



Benutzerhandbuch

SecFlow

Ermittlung sicherheitskritischer Datenflüsse in Quellcode

JK/DR

Version 1.0 / 2008-11-27

Diese Dokument wurde im Rahmen des SecFlow-Projektes durch CC GmbH erstellt.

Ohne ausdrückliche schriftliche Genehmigung von CC GmbH darf kein Teil dieser Unterlagen für irgendwelche Zwecke vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln, elektronisch oder mechanisch, dies geschieht.

CC GmbH übernimmt für diese Unterlagen keine Garantie irgendwelcher Art, einschließlich - aber nicht beschränkt auf - die Nutzbarkeit für den geplanten Einsatzzweck. CC GmbH ist nicht haftbar für Fehler, die in diesen Unterlagen enthalten sein könnten und die daraus ggf. entstehenden Schäden.

Die in diesem Dokument enthaltenen Informationen können ohne vorherige Ankündigung jederzeit geändert werden.

© CC GmbH, Wiesbaden 2008. Alle Rechte vorbehalten.

Inhaltsverzeichnis

1	Zweck und Struktur des Dokuments.....	6
2	Das SecFlow Projekt.....	7
2.1....	Problemstellung und Zielrichtung des Projekts.....	7
2.2....	SecFlow Lizenz.....	8
3	Architektur des Analysesystems.....	10
3.1....	Preprocessing.....	11
3.2....	Sprachkomponente.....	12
3.3....	Adapterkomponente.....	12
3.4....	Summarykomponente.....	13
3.5....	Analysekomponente.....	13
3.6....	Filterkomponente.....	14
3.7....	Präsentationskomponente.....	14
4	Benutzerschnittstelle (Eclipse-Plugins).....	15
4.1....	Komponenten des Plugins.....	15
4.1.1....	Flow-Generator.....	16
4.1.2....	Flow-Editor.....	16
4.1.3....	Konfigurationseditor.....	16
4.1.4....	Präsentationskomponente.....	17
4.2....	Systemvoraussetzungen und Installation.....	17
4.2.1....	Systemvoraussetzungen.....	17
4.2.2....	Installation.....	18
4.3....	Preprocessing.....	22
4.3.1....	PDB2XML.....	22
4.4....	Ausführen der Analyse.....	23
4.4.1....	Konfigurationsdateien.....	23
4.4.2....	SecFlow Project Editor.....	26
4.4.3....	Ausführung.....	29
4.5....	Views.....	29
4.5.1....	Library View.....	29
4.5.2....	Flow View.....	30
4.5.3....	Flow Outline View.....	32
4.5.4....	Result View.....	34
4.5.5....	Path View.....	34
4.6....	Reporting.....	36
5	Konfigurations-/Ausgabe-Dateien.....	39
5.1....	Konfigurations-Datei *.secflow.....	39
5.2....	Flow-Dateien *.flow.....	39
5.2.1....	Syntax einer FLOW-Datei.....	40

5.3....relation.datalog: Definition von Relationen.....	44
5.4....rules.datalog: Definition von Regeln.....	44
5.5....Pfad-Datei *.out.....	45
5.6....Quad und seine Komponenten.....	46
5.6.1....Operatoren in Quad.....	47
5.6.2....Operanden in Quad.....	49
6 Verwendungsbeispiele.....	50
6.1....SQL-Injection.....	50
6.2....XSS – Cross Site Scripting.....	56
7 Glossar.....	61
8 Literatur- und Linksammlung.....	66

Revisionsliste

Kapitel	Inhalt	Datum	verantwortlich
Alle	Ersterstellung	27.11.2008	SecFlow

1 Zweck und Struktur des Dokuments

SecFlow ist ein Werkzeug zur Ermittlung sicherheitskritischer Datenflüsse in Quellcode. Derzeit ist das Tool in einer Beta-Version für die Programmiersprache Java und alle .NET-Sprachen (insbesondere C#) verfügbar. In diesem Dokument werden die grundlegenden Informationen zum Aufbau des Tools und seiner Bedienung wiedergegeben.

Das SecFlow Framework entstand im Rahmen eines Forschungsprojektes im Förderprogramm "Software Engineering 2006" des BMBF. In Kapitel 2 des vorliegenden Benutzerhandbuches werden die Rahmenbedingungen und Hintergründe zur Entstehung von SecFlow kurz dargelegt. Dazu zählt auch die Beschreibung der Motivation und Zielsetzung des Projektes.

Kapitel 3 geht auf die Architektur der Software und die wichtigsten angewandten Prinzipien zur Durchführung der Analyse ein. Es war Vorgabe an die Architektur sieht vor, dass der Softwarekern des Werkzeugs sprachunabhängig funktioniert und durch jeweils sprachabhängige Plugin-Module Software verschiedener Quellsprachen analysiert werden kann. Die Analyse wird dabei auf dem jeweiligen Bytecode (JBC/Java bzw. CIL/.NET) ausgeführt. Der sprachabhängige, stackbasierte Bytecode wird in die registerbasierte, sprachunabhängige Zwischensprache Quad übersetzt. Diese Darstellung wird in weiteren Schritten in verschiedene Datenbankrelationen zerlegt. Aus diesen Relationen wird schließlich der Datenflusspfad abgeleitet.

Die SecFlow-Komponenten wurden in die Eclipse IDE integriert. Die Konfiguration und die Ergebnisdarstellung werden über ein entsprechendes Eclipse-Plugin bereitgestellt. Der Aufbau des Plugins sowie die Funktionen seiner einzelnen Komponenten werden im vierten Kapitel dargelegt.

Wichtige Dateien, die die Konfiguration des Werkzeugs und seine Ausgabe betreffen, werden in Kapitel 5 kurz beschrieben. Der Benutzer erhält Informationen, die die Arbeit mit dem SecFlow-Framework vereinfachen und die Erweiterung und Modifizierung der Analyse bei einer Weiterentwicklung des Tools unterstützen sollen. Dazu zählen u.a. die kurze Darstellung der Zwischensprache Quad sowie die kurze Beschreibung der Datalog-Dateien, die die Analyseregeln beinhalten und deren Inhalte für die Güte und Qualität der Analyseergebnisse maßgeblich verantwortlich sind.

Einige Beispiele zur Anwendung der Analyse mit SecFlow sind in Kapitel 6 zu finden. Diese kurzen Beschreibungen von möglichen Anwendungsfällen sollen dabei helfen, den Analyseverlauf besser nachvollziehen zu können.

Kapitel 7 enthält ein Glossar mit den Erklärungen der wichtigsten fachspezifischen Begriffe, die in diesem Handbuch Verwendung finden oder für die Verständlichkeit der skizzierten Konzepte und Lösungen hilfreich sind.

Zitierte Fachliteratur und interessante Artikel und Webseiten sind im abschließenden Kapitel 8 aufgelistet.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	6 von 67

2 Das SecFlow Projekt

Seit 2003 unterstützt das Bundesministerium für Bildung und Forschung (BMBF) mit der Forschungsinitiative "Software Engineering 2006" Forschungsvorhaben zur Stärkung der Softwaretechnik in Deutschland. Dadurch sollen insbesondere softwareentwickelnde kleine und mittlere Unternehmen (KMU) zu mehr Anstrengungen in der Forschung und Entwicklung angeregt und besser in die Lage versetzt werden, auf Veränderungen rasch zu reagieren und den erforderlichen Wandel aktiv mitzugestalten. Zuwendungen des BMBF sollen Forschungsarbeiten unterstützen, die ohne Förderung nicht durchgeführt werden könnten (vgl. www.softwarefoerderung.de).

2003 erfolgte die erste Auswahlrunde zur Forschungsinitiative. Das SecFlow Projekt ("Security Flow") wurde im Rahmen der zweiten Auswahlrunde zur Forschungsinitiative 2006 in den Kreis der förderwürdigen Vorhaben aufgenommen. Innerhalb der Forschungsinitiative "Software Engineering 2006" des BMBF ist SecFlow dem Themenbereich II – Korrektheit, Sicherheit und Zuverlässigkeit von Software-Systemen (inkl. Safety) – zugeordnet.

SecFlow ist ein Verbundprojekt, an dem insgesamt vier Verbundpartner beteiligt sind:

- Fraunhofer Institut für Experimentelles Software Engineering (IESE), Kaiserslautern
- ICT Solutions AG (ICT), Trier
- SHE Informationstechnologie AG (SHE), Ludwigshafen
- CC GmbH (CC), Wiesbaden

Das Projekt wurde im November 2008 beendet. Ergebnis des Forschungsprojektes ist ein Prototyp des SecFlow-Werkzeuges inklusive sprachspezifischer Plugin-Module für Java und C#/ASP.NET zur Sicherheitsanalyse von Programm-Quelltexten in diesen Programmiersprachen. Er wird als Open Source Software zur Verfügung gestellt.

2.1 Problemstellung und Zielrichtung des Projekts

Je stärker softwarebasierte Systeme in einer vernetzten Welt miteinander gekoppelt werden, desto mehr Angriffspunkte für böswillige Manipulationen entstehen auch und desto weitreichender werden die möglichen Folgen eines solchen Angriffs für das Gesamtsystem. Die Sicherheit von Software gegen Angriffe rückt damit immer stärker ins Zentrum der Software-Entwicklung. Ein wichtiger Ansatz dabei ist der Einsatz von Werkzeugen zum automatischen Aufdecken von Sicherheitsschwachstellen im Quelltext der Software, denn solche Werkzeuge lassen sich, eine problemlose Benutzbarkeit vorausgesetzt, einfach in einen Entwicklungsprozess integrieren und auch für bereits existierende (Legacy-) Software nutzen.

Eine zentrale Klasse von Angriffen auf Software beruht auf vom Angreifer geschickt gewählten Eingabedaten, die ohne ausreichende Prüfung als Parameter für sicherheitskritische Operationen der Software benutzt werden.

Ein Angreifer tippt zum Beispiel in das Benutzernamen-Feld einer Anmeldeseite keinen normalen Namen, sondern einen SQL-Datenbank-Befehl, und die dahinter stehende Anwendung leitet diesen "Namen" ungeprüft an eine Datenbank weiter, die den Befehl dann "unwissentlich" ausführt und dadurch Schaden nimmt. Solche ungeprüften Datenflüsse von Eingaben zu kritischen Operationen müssen in der Software gefunden werden, um sie dann durch Einfügen geeigneter Prüfungen entschärfen zu können. Von Hand ist dies fehleranfällig und teuer. Ziel des SecFlow Projekts ist daher die Entwicklung eines flexiblen und konfigurierbaren Werkzeugs zur Ermittlung ungeprüfter, sicherheitskritischer Kontroll- und Datenflüsse in Quelltext. Neben dem Schutz vor Angriffen (Security) verbessert eine solche Eliminierung ungeprüfter Datenflüsse auch die allgemeine Fehlersicherheit (Safety), da auch viele Fehler in Folge von "unglücklichen" Eingaben mit aufgedeckt werden.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	7 von 67

Das im Projekt entwickelte Framework findet die Datenflüsse in Programm-Quelltexten, die von nicht vertrauenswürdigen externen Datenquellen ("Quellen") zu Aufrufen sicherheitskritischer programmexterner Operationen ("Senken") führen, ohne dass die Daten auf ihrem Weg von der Quelle durch das Programm zur Senke validiert, d.h. auf Konformität mit der vom Programm erwarteten Syntax und Semantik geprüft werden. Das Framework setzt auf Methoden der statischen Analyse auf, d.h. die Analyseergebnisse werden allein durch eine Untersuchung des Quell- bzw. Bytecodes, nicht aber durch Ausführung der zu untersuchenden Software gewonnen.

Die Projektpartner haben sich entschieden, den Schwerpunkt auf Web-Anwendungen zu setzen. Als Ziel-Programmiersprachen wurden Java sowie C# und ASP.NET (Active Server Pages, .NET-Version) realisiert. Auch die weiterführenden Programmierplattformen dieser beiden Sprach-"Welten", J2EE (Java 2 Enterprise Edition) und Microsoft.NET, die für Server-Anwendungen in der Praxis meist unverzichtbar sind, können in gleicher Weise analysiert werden.

Bei der Realisierung des Werkzeuges wurden Teile verschiedener Open Source Software im Framework adaptiert und integriert. Zu nennen sind dabei vor allem Joeq [8] und BDDBDDB [7]. Joeq ist ein Java-Compiler-Framework inklusive einer Java Virtual Machine. Die virtuelle Maschine betreffenden Teile wurden aus dem Framework entfernt. Weitere Anpassungen von Joeq betreffen hauptsächlich die Fähigkeit zur Analyse mehrerer Programmiersprachen.

Die Analyse des Daten- und Kontrollflusses wurde in der logisch-relationalen Programmiersprache Datalog implementiert. Zur Ausführung des Datalog-Codes wird die deduktive Datenbank BDDBDDB – Binary Decision Diagrams Based Deductive DataBase – benutzt [10]. Die Architektur von BDDBDDB wurde erweitert, so dass Daten aus beliebigen Quellen gelesen werden können.

Um eine relativ einfache Bedienbarkeit der Software sowie die Darstellung der Ergebnisse in einer leicht verständlichen Art und Weise zu erreichen, wurden die SecFlow-Komponenten in Eclipse als weit verbreitetes und anerkanntes Entwicklungssystem integriert.

2.2 SecFlow Lizenz

SecFlow wird als freie Software zur Verfügung gestellt und steht unter der GNU GPL General Public License, Version 3 vom 29. Juni 2007. Es darf damit unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergegeben und modifiziert werden, entweder gemäß Version 3 der Lizenz oder jeder späteren Version.

Eine Kopie der GNU License wird mit SecFlow zusammen ausgeliefert. Sie befindet sich in der Datei COPYING.txt. Sollte diese Datei nicht auffindbar sein, wird auf <http://www.gnu.org/licenses/> verwiesen.

```
SecFlow - Unsecure dataflow detection and analyzation framework
Copyright © 2008 SecFlow, www.secflow.org
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>
```

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	8 von 67

Das SecFlow Framework beinhaltet und verwendet freie Software -- zum Teil auch modifizierte Teile – anderer Hersteller. Diese Software wird entsprechend der und unter den originären Lizenzbedingungen der Hersteller weitergegeben. Im einzelnen betrifft dies folgende Produkte und Lizenzen:

Produkt	Lizenz
Joeq	LGPL
JavaBDD	LGPL
bddbddb	LGPL
Eclipse	EPL
Mbel	LGPL
Jasper/iReport	APACHE2.0
pdb2xml	Mike Stall

Die genannten Produkte sind mit ihren Lizenzbedingungen in der Datei THIRDPARTYLICENSEREADME.txt aufgeführt. Sollte diese Datei nicht auffindbar sein, wird an den jeweiligen Hersteller verwiesen.

3 Architektur des Analysesystems

Das SecFlow-Werkzeug dient der Analyse von Software auf eingabebasierte Sicherheitsschwachstellen. Die Architektur sieht vor, dass Software verschiedener Quellsprachen analysiert werden kann. Aktuell unterstützt das Werkzeug die Programmiersprachen Java und C#, wobei jeweils der Bytecode (JBC/Java bzw. CIL/.NET-Assembly) analysiert wird. Der Zwischencode muss bei der Generierung vom Compiler mit Debuginformationen angereichert werden. Dieser erste Schritt, der nicht zum SecFlow-Werkzeug gehört, sondern von konventionellen Compilern erledigt wird, wird unter der Bezeichnung "Preprocessing" zusammengefasst.

Die Sprachkomponente (Language Component) übersetzt den stackbasierten, sprachabhängigen Bytecode in die registerbasierte, sprachunabhängige Zwischensprache Quad (siehe unten Abschnitt 3.2 und Kap. 5.6). Der so entstehende Zwischencode dient als Schnittstelle zu nachfolgenden SecFlow-Komponenten. Er beinhaltet die Datenbasis für die spätere Analyse.

Die Adapterkomponente (Adapter Component) übersetzt den Zwischencode in relationale Tupel. Diese Tupel bilden im Quadcode enthaltene Zusammenhänge ab.

Datenfluss-Summaries sind Modelle externer Funktion, die nicht direkt analysiert werden können. Sie müssen manuell spezifiziert werden. Die Summarykomponente (Summary Component) übersetzt die Datenfluss-Summaries und macht sie damit für das Werkzeug verfügbar.

Die Analysekomponente (Analysis Component) führt die eigentliche Datenflussanalyse durch und berechnet aus den zuvor generierten Tupeln und den von der Summarykomponente übersetzten Datenfluss-Summaries eine Rohform der kritischen Pfade. Das Ergebnis der Analysekomponente sind verschiedene Datenbankrelationen, welche tupelweise ausgelesen werden können.

Die Filterkomponente (Filter Component) entwickelt aus den Ergebnisrelationen der Analysekomponente einen Graphen und extrahiert aus diesem Graphen einen redundanzfreien Datenflusspfad. Der Pfad wird in XML-Format ausgegeben. In der Präsentationskomponente (Presentation Component) können die Pfade, die im XML-Format ausgegeben wurden, visualisiert werden.

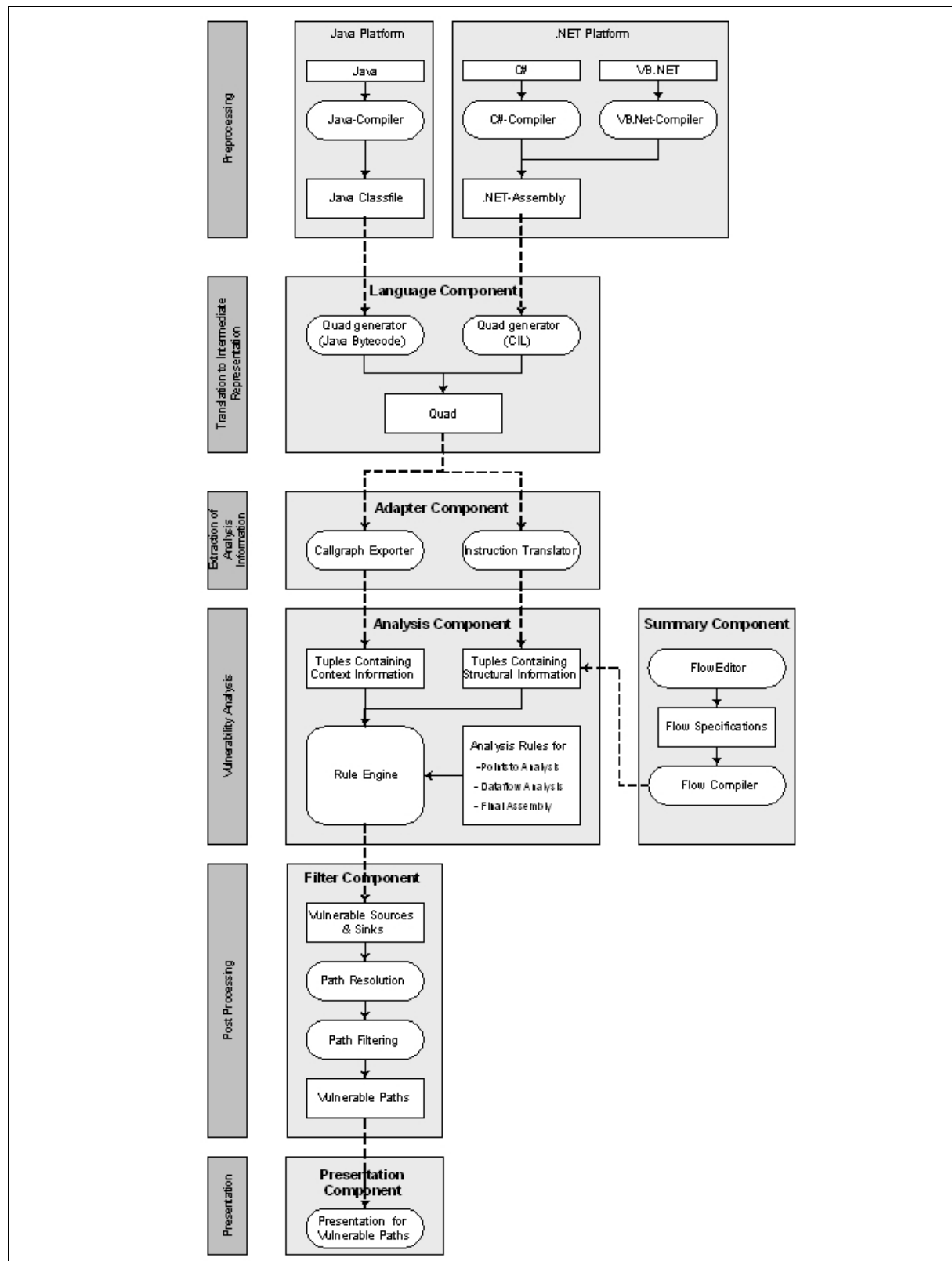


Abb.: Architektur des Analysesystems

3.1 Preprocessing

Der sprachabhängigen Analyse vorgeschaltet sind die Preprocessing-Schritte. Um die Unschärfe bei der Quellcodeübersetzung mit Parser zu vermeiden wurde im Fall von Java und .NET entschieden, Bytecode aus kompilierten Quellen als Grundlage zu verwenden. Beide Sprachen sind so konzipiert, dass sie den Quellcode zunächst in einen generischen universellen Bytecode übersetzen. Die Spezifikation des Bytecodes ist weitestgehend starr und robust gegen Änderungen in der Quellsprache.

Eingelesen werden im Fall von Java die .class-Dateien, im Fall von .NET die .dll- und .exe-Dateien. Diese Dateien werden zuvor mit einem konventionellen Java- bzw. .NET-Compiler aus dem Quellcode generiert. In beiden Fällen muss zusätzlich Debuginformation bereit gestellt werden, damit später die Zeilenrückverfolgung gewährleistet ist. Java stellt diese Debuginformation direkt im Bytecode dar. Bei .NET sind sie in der Program Database (.pdb) als separate Datei verfügbar.

3.2 Sprachkomponente

Der Kern der Sprachkomponente entstand aus dem Joeq-Compiler-Framework [8]. In Joeq ist eine Java Virtual Maschine integriert. Die JVM bzw. die sie betreffenden Teile wurden aus dem Framework herausgelöst, um es auf die für SecFlow notwendige Funktionalität zu beschränken und es besser an die Bedürfnisse von SecFlow anpassen zu können. Die Anpassung von Joeq wurde hauptsächlich hinsichtlich der Fähigkeit zur Analyse mehrerer Programmiersprachen durchgeführt. Die Sprachkomponente übersetzt die durch das Preprocessing gewonnene Programmdarstellung in einen Zwischencode in der Sprache Quad. Quad enthält nur primitive, weitgehend atomare Operationen und ist somit geeignet, verschiedene Quellsprachen abzubilden. Für .NET und Java existieren zwei unabhängige Sprachplugins (Quad Generatoren), die die Transformation von Bytecode (bzw. .NET-Assembly) in Quadcode durchführen. Die typische Funktionalität eines Sprachplugins wird nachfolgend am Beispiel des Java-Plugins charakterisiert:

- Die Sprachkomponente parst den Bytecode und entwickelt daraus einen Abstrakten Syntaxbaum (AST).
- Die Sprachkomponente parst den Bytecode der Typen, die direkt oder indirekt von den bereits vorhandenen ASTs referenziert werden.
- Die Sprachkomponente erzeugt Quadcode, der die SSA(Static Single Assignment)-Eigenschaft erfüllt, d.h. dass jede im Quadcode verwendete Variable nur an genau einer Stelle zugewiesen wurde.
- Die Sprachkomponente stellt alle erzeugten Informationen (nachfolgend als Codegraph bezeichnet) über die API zur Verfügung

3.3 Adapterkomponente

Die Adapterkomponente überführt die Informationen im Codegraph in ein deduktives Datenbankmodell. Die Adapterkomponente schreibt alle Relationen platzsparend als Binary Decision Diagrams (BDDs) auf die Festplatte. Dazu traversiert der Instruction Translator den Codegraph. An jedem Knoten im Codegraph, an dem analyserelevante Zusammenhänge beschrieben werden, emittiert die Adapterkomponente ein passendes Relationentupel (Structural Information) (Hinweis: Der Begriff *Relation* wird für die Bezeichnung von Datalog-Prädikaten verwendet). Beispiel:

- Die Adapterkomponente traversiert einen Klassenknoten. Folgende Knoten hängen mit diesem Klassenknoten zusammen
 - Knoten für Sichtbarkeit und Zugriffskontrolle (public, protected, private, static, abstract)
 - Methodenknoten
 - Instanzvariablenknoten
 - Obertypknoten
- Jeder Zusammenhang wird als Tupel einer passenden Relation emittiert:

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	12 von 67

- (Attribut, Klasse) in classattributes
- (Methode, Klasse) in methodsofclass
- (Instanzvariable, Klasse) in fieldsofclass
- (Oberklasse, Klasse) in superclasses

In einem zweiten Schritt traversiert der Callgraph Exporter den Codegraph und analysiert dabei die Aufrufbefehle, im speziellen die dort gefundenen Beziehungen zwischen Aufrufstelle und Aufrufziel. Diese werden zunächst in Form eines Callgraphs angeordnet. Dieser Callgraph wird dann ähnlich dem Verfahren von Lam/Whaley [5] in eine Kontext-Relation umgewandelt, die später bei der kontextsensitiven Points-to-Analyse benötigt wird (siehe dazu unter 3.5 Analyse-Komponente).

3.4 Summarykomponente

Eine Forderung an das SecFlow-Werkzeug stellt die Sicherstellung der Analysierbarkeit einer Anwendung dar, auch dann, wenn nicht alle Teile dieser Anwendung im Analyseumfang enthalten sind. Insbesondere soll auch darauf verzichtet werden können, Bibliotheken von Drittanbietern in den Analyseumfang einzubeziehen. Voraussetzung dafür ist, dass alle nicht zu analysierenden Teile des Programms durch Datenfluss-Spezifikationen (Summaries bzw. Flow Spezifikationen) beschrieben wurden. Die Spezifikation von Summaries erfolgt in der Datenfluss-Spezifikationssprache Flow (siehe Kap. 5.2). Diese Sprache, die sich an die Syntax moderner objektorientierter Sprachen anlehnt, drückt aus, welche Datenflüsse von einer speziellen Methode erzeugt werden. Sie kann also zur kompakten Charakterisierung der relevanten Datenfluss-Eigenschaften des eigentlichen Codes dienen, ohne dass der Code selbst in die weiteren Analysen einbezogen werden muss. Dateien in der Sprache Flow können mit einem eigens dazu entwickelten Flow-Editor bequem erzeugt und geändert werden. Der Flow-Compiler übersetzt (manuell zu erstellende) Summaries in Tupel, wie sie von der Analysekomponente benötigt werden. Flow enthält keine Funktionsaufrufe und somit entfällt auch die Behandlung des Callgraphs.

3.5 Analysekomponente

In der Analysekomponente laufen die Tupel aus der Adapterkomponente und der Summarykomponente zusammen.

Die Analyseregeln wurden in der logisch-relationalen Programmiersprache Datalog implementiert. Der Kern dieser Komponente ist ein Regelinterpreter (Rule Engine), der eine Deduktive Datenbank verwaltet. Es handelt sich hierbei um eine Weiterentwicklung des Werkzeugs BDDBDD [10]. Wie jede Datenbank besteht auch diese Deduktive Datenbank aus einem Schema und Daten. Das Schema ist unterteilt in

- Relationsdefinitionen (relations.datalog): entspricht Tabellendefinitionen in SQL-Datenbanken (siehe weiter Kap. 5.3)
- Regeldefinitionen (rules.datalog): entspricht Views in SQL-Datenbanken (siehe weiter Kap. 5.4)

Mit diesem Regelinterpreter werden folgende Schritte nachvollzogen:

- Berechnung der Points-to-Information (welche Variable/welcher Pfad zeigt auf welches Objekt) durch Nachvollziehen des Variablenflusses
- Berechnung der kontextsensitiven Methodenaufrufe durch Verknüpfung der Methodenaufrufe mit ihrer Kontextinformation
- Übertragung der Kontextinformationen vom Methodenaufruf auf die aufgerufene Methode und ihre Operationen

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	13 von 67

- Synthese des Datenflusses durch Interpretation der nun kontextsensitiven Operationen unter Berücksichtigung von datenfluss-unterbrechenden Operationen (Sanitizer, Guards)

Das Ergebnis dieser Analyse besteht aus folgenden Relationen:

- `untrusted`: verdächtige Quellen im Quadcode
- `vulnerable`: verdächtige Senken im Quadcode
- `criticalpath`: Quellen und Senken, die miteinander verbunden sind
- `visiblecriticalelements`: Elemente auf einem `criticalpath`
- `usedregister2expression`: Übersetzung von Quadregister nach Quellcode-Ausdruck
- `usedlocation2line`: Übersetzung von Quadcodeindex nach Zeilennummer

Das Zusammenfügen dieser Fragmente zu Datenflusspfaden übernimmt die Filterkomponente.

3.6 Filterkomponente

Die Filterkomponente fügt die von der Analysekomponente erzeugten Relationentupel (Vulnerable Sources and Sinks) zu einem Quadcodepfad zusammen, übersetzt den Quadcodepfad in Quellcodenotation (Path Resolution) und eliminiert triviale oder redundante Pfadsequenzen (Path Filtering). Der Eliminierungsschritt ist notwendig, weil verschiedene Quad-Variablen gleiche Quellcode-Ausdrücke darstellen können. Da der Anwender Quellcode bevorzugt, würde eine Direktübersetzung zweier solcher Variablen zur Folge haben, dass der gleiche Ausdruck mehrfach ausgegeben wird. Das Endergebnis der Filterung ist eine Menge duplikatfreier, als kritisch eingestufte Pfade (Vulnerable Paths), abgelegt im XML-Format (Datei *.out). In dieser Datei sind Informationen zu der Analyse, dem Pfad und seinen Elementen abgelegt. Die XML-Datei ist im Kapitel 5.5 weiter beschrieben.

3.7 Präsentationskomponente

Die Präsentationskomponente sorgt für eine benutzerfreundliche Aufbereitung der Analyseergebnisse aus der von der Filterkomponente bereitgestellten Ausgabedatei im XML-Format. Die verwundbaren Pfade (vulnerable paths) werden übersichtlich dargestellt und im Quelltext der analysierten Software markiert. Darüber hinaus können Reports in verschiedenen Formaten generiert werden.

Die Präsentationskomponente ist Bestandteil der SecFlow-Benutzerschnittstelle, die als Eclipse-Plugin realisiert wurde. Die Beschreibung erfolgt im nachfolgenden Kapitel 4.

4 Benutzerschnittstelle (Eclipse-Plugins)

Das bereitgestellte Eclipse-Plugin stellt die Benutzeroberfläche für das SecFlow Framework dar. Es bietet Funktionen zur Konfiguration des Analyseprojekts, zur Ausführung und Ergebnisdarstellung sowie Funktionen zur Bearbeitung von Summary Dateien (auch Flow Dateien genannt) zur Abbildung externer Funktionen.

4.1 Komponenten des Plugins

Die SecFlow-Komponenten wurden in der Eclipse IDE integriert. Die Konfiguration und die Ergebnisdarstellung werden über ein entsprechendes Eclipse-Plugin bereitgestellt. Dieses besteht aus einem Flow-Generator zur Generierung von Template Summary Dateien, einem Flow Editor zur Bearbeitung von Summary Dateien, einem Konfigurationseditor zur Festlegung der Eingabeparameter und zur Ausführung der Analyse, sowie einer Reporting-Komponente zur Ergebnisdarstellung.

Das Eclipse Plugin wird über eine speziell für SecFlow entwickelte Eclipse Update Site bereitgestellt.

Einen Überblick über die einzelnen Komponenten und ihre Zusammenhänge bietet die nachfolgende Abbildung:

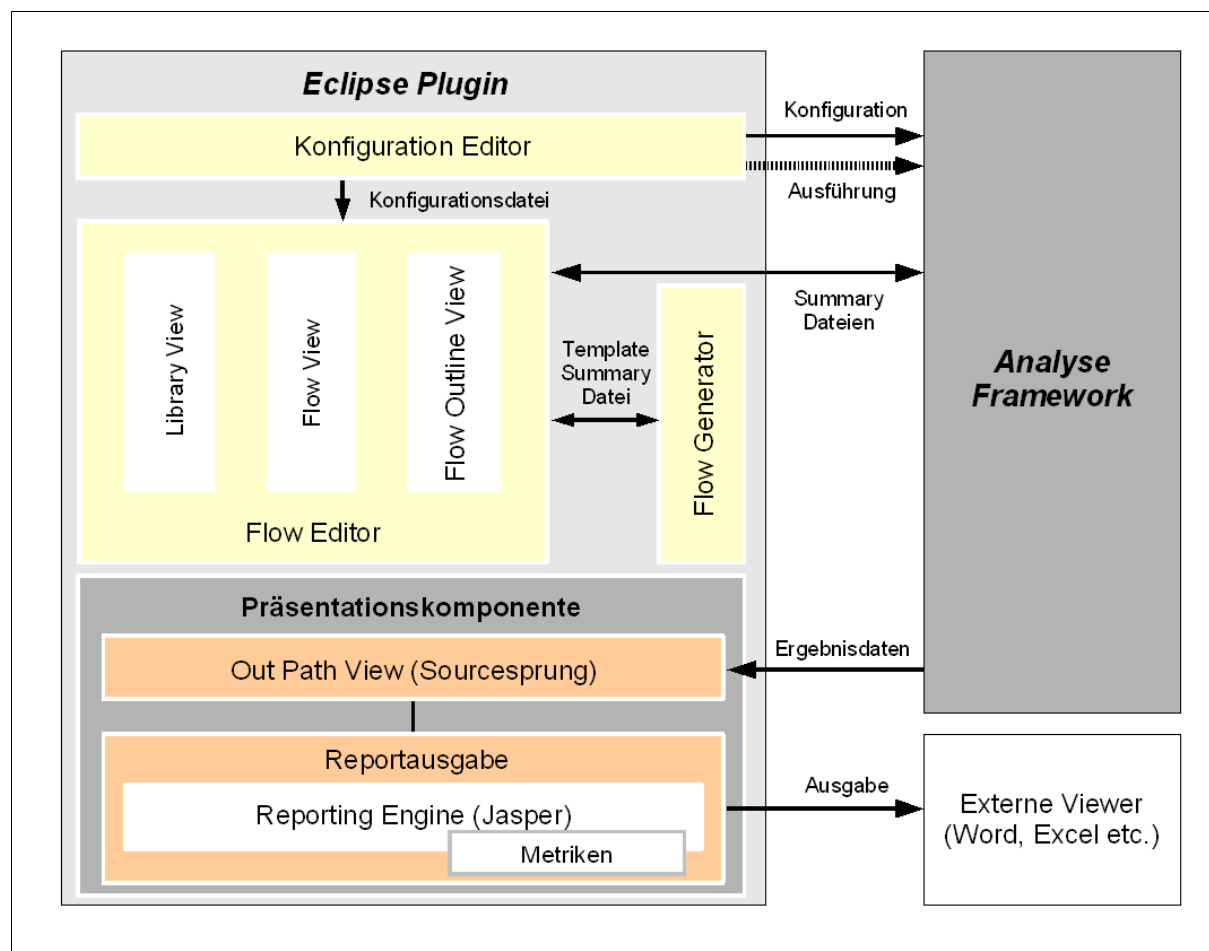


Abb.: Aufbau des Eclipse-Plugins

4.1.1 Flow-Generator

Summary Dateien (auch Flow Dateien genannt) sind eine abstrakte Darstellung von Bibliotheksfunktionen und den dort vorhandenen Datenflüssen. Sie steigern die Analyseperformance, indem sie komplexe Datenflüsse von Bibliotheksfunktionen komprimiert abbilden und ermöglichen es, Quellen und Senken innerhalb von Bibliotheksfunktionen (Ein- und Austrittspunkte eines kritischen Pfades) festzulegen. Werden Methoden nicht abgebildet oder nicht richtig abgebildet, kann dies dazu führen, dass Datenflüsse nicht analysiert werden können. Kritische Informationen können somit verloren gehen.

Summary Dateien (= Flow Dateien) sind Textdateien, die von dem Summary-Parser, einem durch ANTLR nach der Summary-Spezifikation generierten Parser, eingelesen und in ein für das SecFlow-Framework verarbeitbares Objekt-Mapping gewandelt werden.

Für das geplante Eclipse-Plugin ergab sich die Forderung, dass es die Struktur von Summary Dateien darstellen, sowie syntaktische Fehler in diesen erkennen kann. Zusätzlich gibt es die Möglichkeit, für ein Entwicklungsprojekt die gültigen, die fehlenden und auch die unvollständigen Summary Dateien anzuzeigen und deren Erstellung bzw. Bearbeitung zu unterstützen.

Mit dem Flow-Generator ist es möglich, unter Angabe der Quelldatei automatisch Templates für Summary Dateien zu erzeugen. Diese beinhalten ein an die Gegebenheiten die jeweiligen Bibliotheksfunktion angepasstes Grundgerüst zur Darstellung des entsprechenden Datenflussmodells. Diese Grundgerüst kann dann vom Anwender mit den entsprechenden Datenflussinformationen gefüllt werden. Datenflüsse selbst werden damit nicht generiert.

4.1.2 Flow-Editor

Der Flow-Editor beinhaltet drei verschiedene Eclipse Views:

Die **Library View** zeigt die in einem Projekt verfügbaren Bibliotheksklassen an. Über die Library View ist es möglich, den Flow-Generator aufzurufen und neue Summary Dateien zu generieren.

Die **Flow View** zeigt die für ein Projekt verfügbaren Summary Dateien. Weiterhin bietet sie Funktionen, um fehlende und unvollständige Summary Dateien (bezogen auf Methoden) im aktuellen Projekt zu bestimmen.

Die **Flow Outline View** zeigt die Objektstruktur einer zur Bearbeitung im Eclipse Editor geladenen Summary Datei. Sie dient der Darstellung

- der Klassenabhängigkeiten,
- der lokalen typenlosen Variablen,
- der verwendeten Methoden und deren Parameter sowie
- von Quellen und Senken.

Die Views werden in Kapitel 4.5 näher beschrieben.

4.1.3 Konfigurationseditor

Der Konfigurationseditor dient zur Konfiguration der Analyse, dem Management der zu analysierenden Quellen sowie der Organisation der benötigten Ressourcen. Dazu wird eine entsprechende Konfigurationsdateien erstellt.

Über den Konfigurationseditor wird die Ausführung der Analysen gestartet.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	16 von 67

4.1.4 Präsentationskomponente

Dem Anwender werden zwei Werkzeuge zur Darstellung der SecFlow Ergebnisse zur Verfügung gestellt.

Die **Path View** bereitet die Ausgaben des SecFlow Werkzeugs auf und stellt sie dem Entwickler in optisch und logisch aufbereiteter Form dar. Aus ihr kann der Anwender die ermittelten sicherheitskritischen Pfade und deren Elemente ablesen. Die View stellt die Schnittstelle zwischen der SecFlow Ausgabe und dem analysierten Quelltext dar. Die angezeigten Pfadelemente sind mit den Quelltextabschnitten verknüpft und ermöglichen es, in die betroffene Codezeile in der analysierten Source zu navigieren.

Die angezeigten Informationen beschränken sich auf die für die Anzeige im Quelltext relevanten Inhalte, wie zum Beispiel Element, Identifier oder Zeilennummer. Darüber hinaus können auch weitere Statistiken bezüglich der Ausgabe des SecFlow Werkzeugs ermittelt werden. Diese kann der Anwender über die Verwendung der **Reporting Engine** erhalten.

Aus der Path View heraus ist es möglich, Reports in HTML (Hypertext Markup Language), PDF (Portable Document Format), ODT (OpenDocument Text) und XLS (Excel Spreadsheet) zu erstellen. Ein Report fasst den analysierten Quelltext und die Ausgabe des SecFlow Werkzeugs zu einem Dokument zusammen und macht einzelne Referenzen zwischen Quelltext und SecFlow-Ausgabe optisch kenntlich.

Weitere Informationen zur Path View sind in Kapitel 4.5.3, zum Reporting in Kapitel 4.6 zu finden.

4.2 Systemvoraussetzungen und Installation

4.2.1 Systemvoraussetzungen

Für die Installation und Verwendung von SecFlow wird folgende Systemumgebung empfohlen:

Betriebssystem:	Windows XP 32-bit, Linux, oder höher (Java 1.6 muss unterstützt werden) Da das Framework sehr speicherlastig arbeitet, wird der Einsatz eines 64-bit OS empfohlen.
CPU/Hauptspeicher:	1GB oder mehr
Speicherplatz:	ca. 40MB plus Speicherplatz für Ausgabedateien
Software:	Java SE Development Kit (JDK), Version 5 oder höher Eclipse 3.4.1 Build id: I20080911-1700 oder höher

Hinweis: Die Speicheroptionen von Eclipse sollten in der Datei "eclipse.ini" im Eclipse-Homeverzeichnis wie folgt angepasst bzw. durch höhere Werte ersetzt werden:

```
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256M
-framework
plugins\org.eclipse.osgi_3.4.2.R34x_v20080826-1230.jar
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms256m
-Xmx1024m
```

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	17 von 67

Von SecFlow können optional Reports in verschiedenen Ausgabeformaten (*.pdf, *.html, *.odt, *.xls) generiert werden. Für die Anzeige in einem bestimmten Format werden ggf. zusätzliche Programme benötigt, die nicht im Lieferumfang von SecFlow enthalten sind.

4.2.2 Installation

Das SecFlow Eclipse-Plugin wird als Eclipse-Updatesite für die Installation zur Verfügung gestellt. Die nachfolgende Installationsbeschreibung basiert auf Eclipse 3.4.1 und beschreibt u.a. das Standardvorgehen zur Updatesite-Installation. Bei der Verwendung anderer Versionen von Eclipse kann es daher zu Abweichungen in einzelnen Installationsschritten kommen.

Für die Installation muß Eclipse gestartet werden. Über den Menüpunkt "Help/Software Updates" wird das Fenster "Software Updates and Add-ons" geöffnet. In der Registerkarte "Available Software" den Button "Add Site" bestätigen.

Im nachfolgenden Dialogfenster muß über "Local" der Installationspfad File:/....\net.secflow.tools.update.site gesetzt werden. Der Installationspfad muss die Ordner "plugins" und "features" enthalten. Nach der Auswahl muss unter "Available Software" der hinzugefügte Pfad und die Unterknoten markiert werden. Die Installation wird mit dem Button "Install..." gestartet. Im Anschluss sollte Eclipse neu gestartet werden.

Hinweis: Bei der Installation müssen die Lizenzbedingungen von SecFlow akzeptiert werden. Nähere Informationen zu den Lizenzbedingungen finden Sie in Kap. 2.2.

Bei erfolgreicher Installation ist das Eclipse-Menü um den Punkt "SecFlow" erweitert. Zudem sind das SecFlow Symbol und die Plugin-Eigenschaften unter dem Menüpunkt "Unter Help| About Eclipse SDK" zu finden.

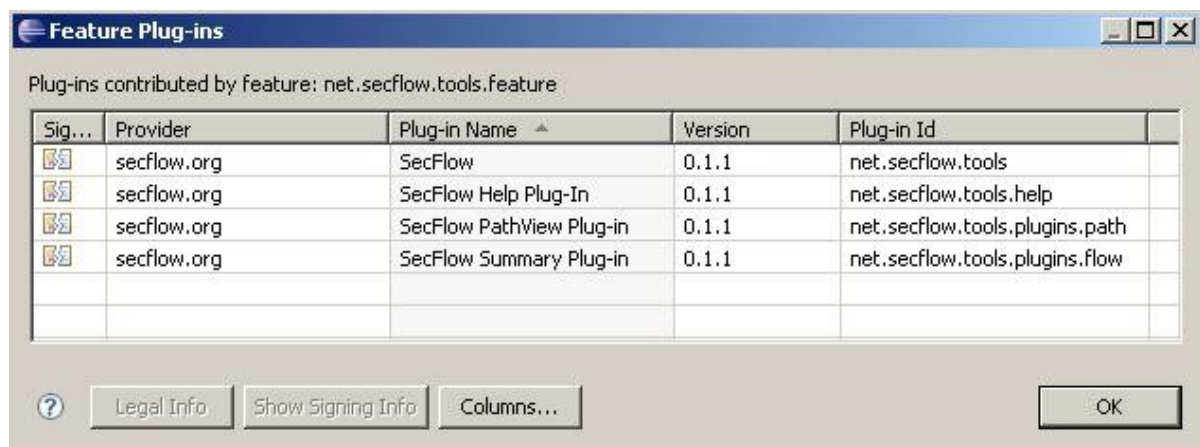


Abb.: Plugin-Eigenschaften

Über den Menüpunkt "Window|Preferences|Secflow|Framework" werden die Verzeichnisse für die Datalog-Dateien sowie für die Dateien mit den Spezifikationen der potentiellen Gefahrenquellen für Sources und Sinks angegeben (siehe Abb. Framework-Einstellungen).

Werden keine Verzeichnisse ausgewählt, so werden die Dateien automatisch in einem temporären Verzeichnis abgelegt.

Button	Beschreibung
1	Verzeichnis für die Datalogdateien
2	Auswahl des Projektverzeichnisses "net.secflo.control/ressources/generated" ACHTUNG: Bei Auswahl dieses Felde müssen die Datalog-Dateien in dieses Verzeichnis kopiert werden!
3	Verzeichnis der Sourcen-Datei consideredsources.conf
4	Verzeichnis der Sink-Datei consideredsinks.conf

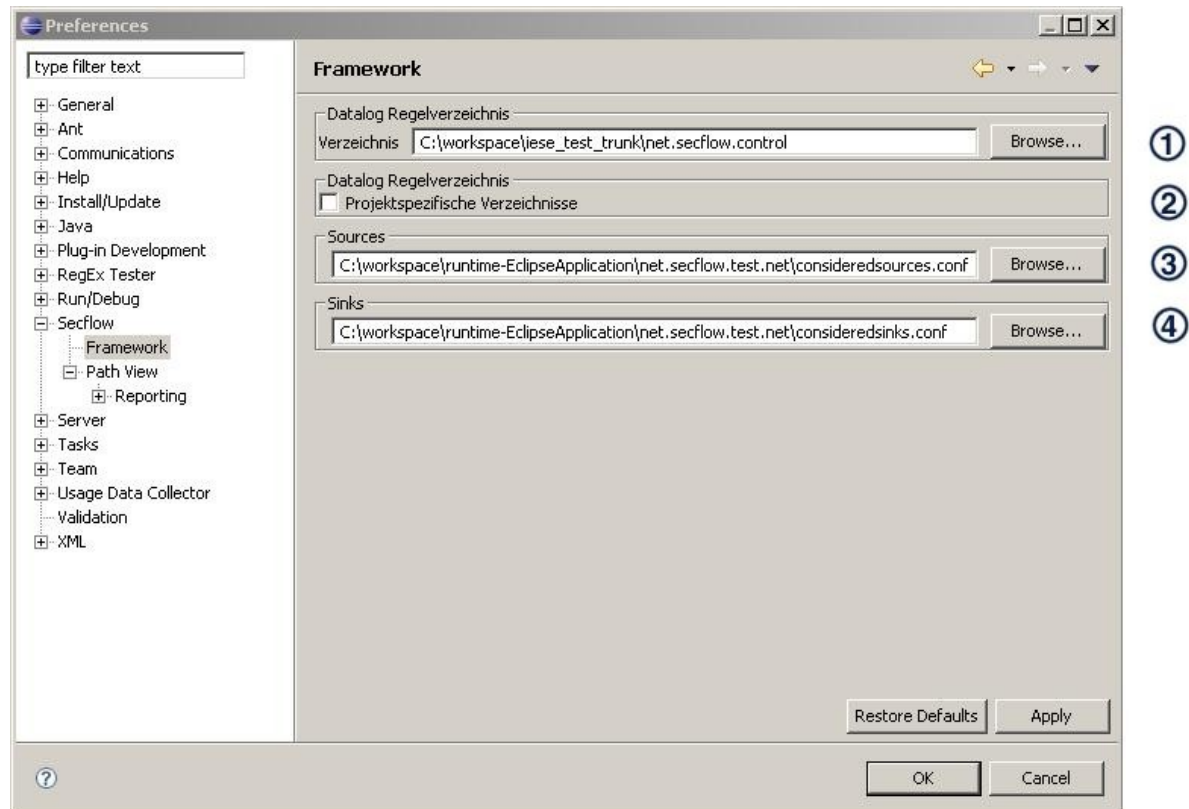


Abb.: Framework-Einstellungen

Der Pathout-Ordner für die auszugebenden Pfaddateien muss in Eclipse über den Menüpunkt "Window|Preferences|Secflow|Path View" unter "Out File Settings|Output Path of out Files" eingetragen werden. Hier sollte "resource/out" als Pfad angegeben werden. In der PathView werden dann immer alle Ausgabepfade im Workspace durchscant und die enthaltenen Dateien angezeigt.

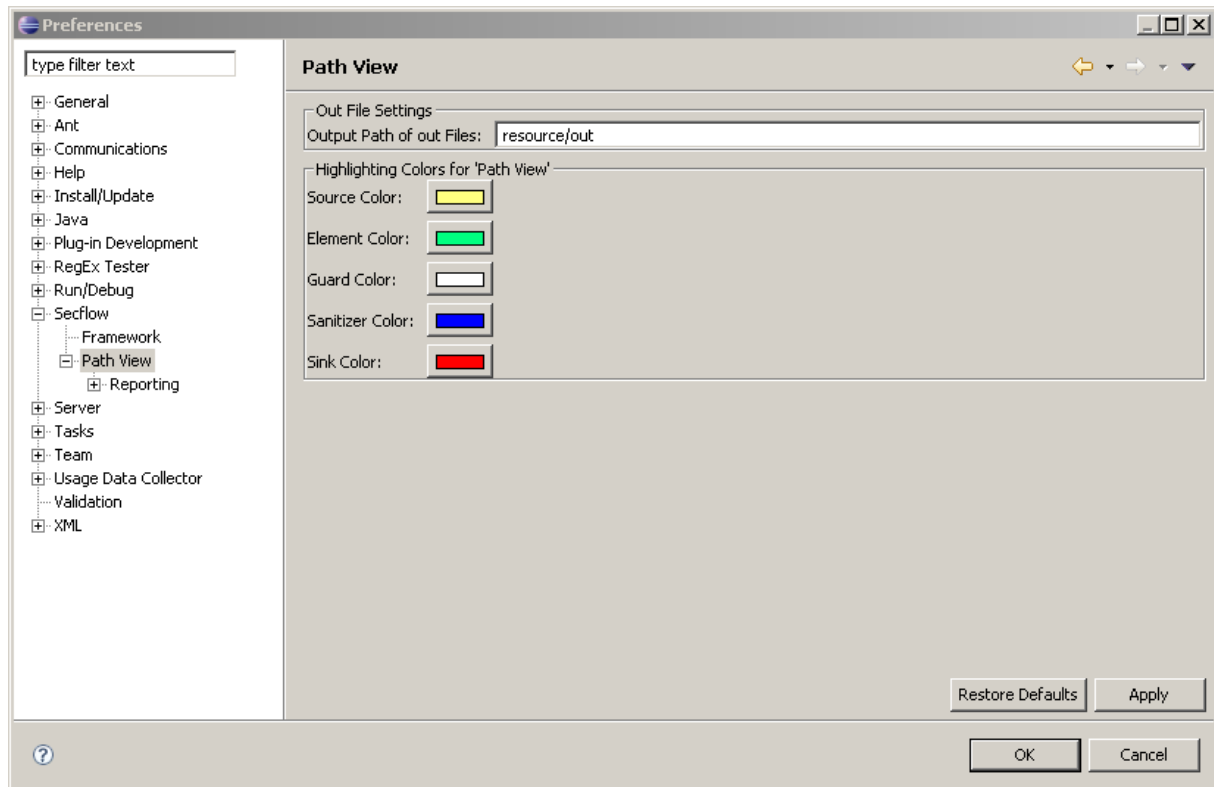


Abb.: Einstellung zum Ausgabepfad

Die Symbole für die Reportgenerierung in der Path View werden erst aktiv, wenn die entsprechenden externen Viewer über den Menüpunkt "Window|Preferences|Secflow|Path View|Reporting" aktiviert wurden.

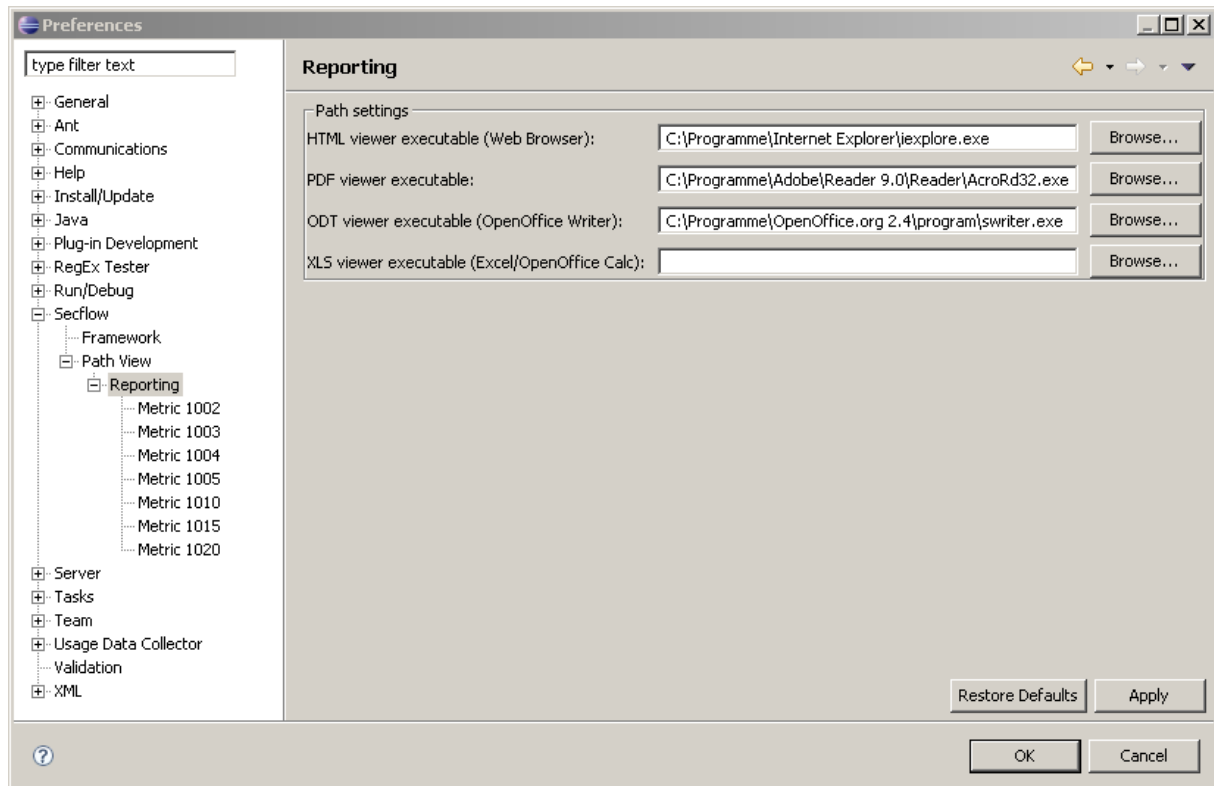


Abb.: Externe Viewer - Pfadangaben

4.3 Preprocessing

4.3.1 PDB2XML

Die SecFlow-Analyse erlaubt die Rückreferenzierung der Ausgabeelemente in den Sourcecode. Unter .NET existiert das Problem, das nur die MS .NET-Compiler ausreichende Debuginformationen mit Sourcecode-Zeilenummern generieren. Die Debuginformationen von Compilern wie Mono reichen nicht dazu aus um eine Rückreferenzierung zu garantieren. Die Zeileninformationen werden über eine XML-Datei mit den nötigen Daten zur Verfügung gestellt. Diese XML-Datei wird aus den durch einen MS .NET-Compiler generierten pdb und Assembly-Dateien generiert. Dazu wurde das von Mike Stall realisierte Tool `pdb2xml` (<http://blogs.msdn.com/jmstall>) erweitert. Das Tool ist nur unter MS Windows ausführbar und zu analysierende Projekte sind entsprechend vorzubereiten. Fehlende oder falsche Bibliotheken und pdb-Dateien werden vom Programm erkannt und angezeigt.

```
pdb2Xml <input dll/exe> <output xml> <.net-path>
* <input dll/exe>  Assembly-Datei mit Suffix dll oder exe
* <output xml>     Name der Ausgabedatei
* <.net-path>      .NET Bibliothekspfade getrennt durch Semikolon
```

Das folgende Bild zeigt das `pdb2xml`-Tool im Kontext des Preprocessings.

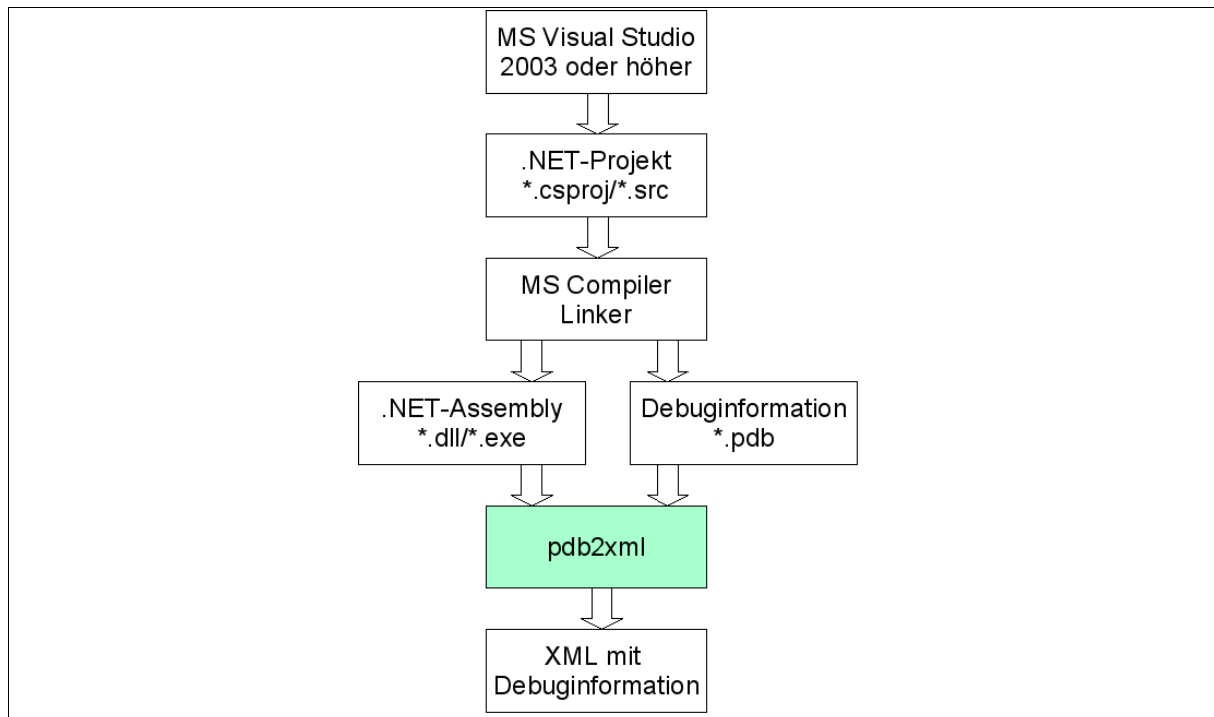


Abb.: `pdb2xml` im Kontext des Preprocessings

4.4 Ausführen der Analyse

Zur erfolgreichen Durchführung einer SecFlow-Analyse muß folgendes Vorgehen beachtet werden:

1. Die Auswahl eines geeigneten Projektes inklusive Sourcen und Debuginformationen
2. Import des Projektes in den Eclipse Workspace
3. Konfiguration einer Analyse mittels SecFlow-Konfigurationseditor
4. Generierung und Bearbeitung von fehlenden oder unvollständigen Flow Dateien
5. Ausführen der Analyse
6. Anzeige der verwundbaren Pfade (sicherheitskritische Pfade) in der Path View und Reports

In den nachfolgenden Kapiteln werden die Schritte 3-6 detailliert dargestellt.

4.4.1 Konfigurationsdateien

Die SecFlow-Analysen wurden ursprünglich mittels JUnit-Tests ausgeführt. Um dem Anwender mehr Flexibilität und eine bessere Übersicht zu gestatten, wurden Konfigurationsdateien eingeführt. Mit diesen ist es möglich, eine Analyse genauer zu spezifizieren.

Anlegen einer Konfigurationsdatei

Die Konfigurationsdatei kann über das Kontextmenü im Package- oder Navigation-Explorer von Eclipse angelegt werden. Dazu wird der jeweilige Projektkopf ausgewählt und das Kontextmenü mittels Rechtsklick geöffnet. Im Kontextmenü wird der Menüpunkt New|Other...|SecFlow gewählt. Es öffnet sich der Dialog "New" mit dem Ordner "SecFlow". Dieser enthält die 2 Optionen ".NET Configuration" und "Java Configuration". Die erstere erzeugt eine leere Datei mit dem Suffix ".clr.secflow" für die Analyse von .NET-Projekten, die letztere erzeugt eine Datei mit dem Suffix ".jvm.secflow" für die Analyse von Java-Projekten. Nach Auswahl einer der beiden Konfigurationen öffnet sich ein Dialog zur Vergabe des Dateinamens.

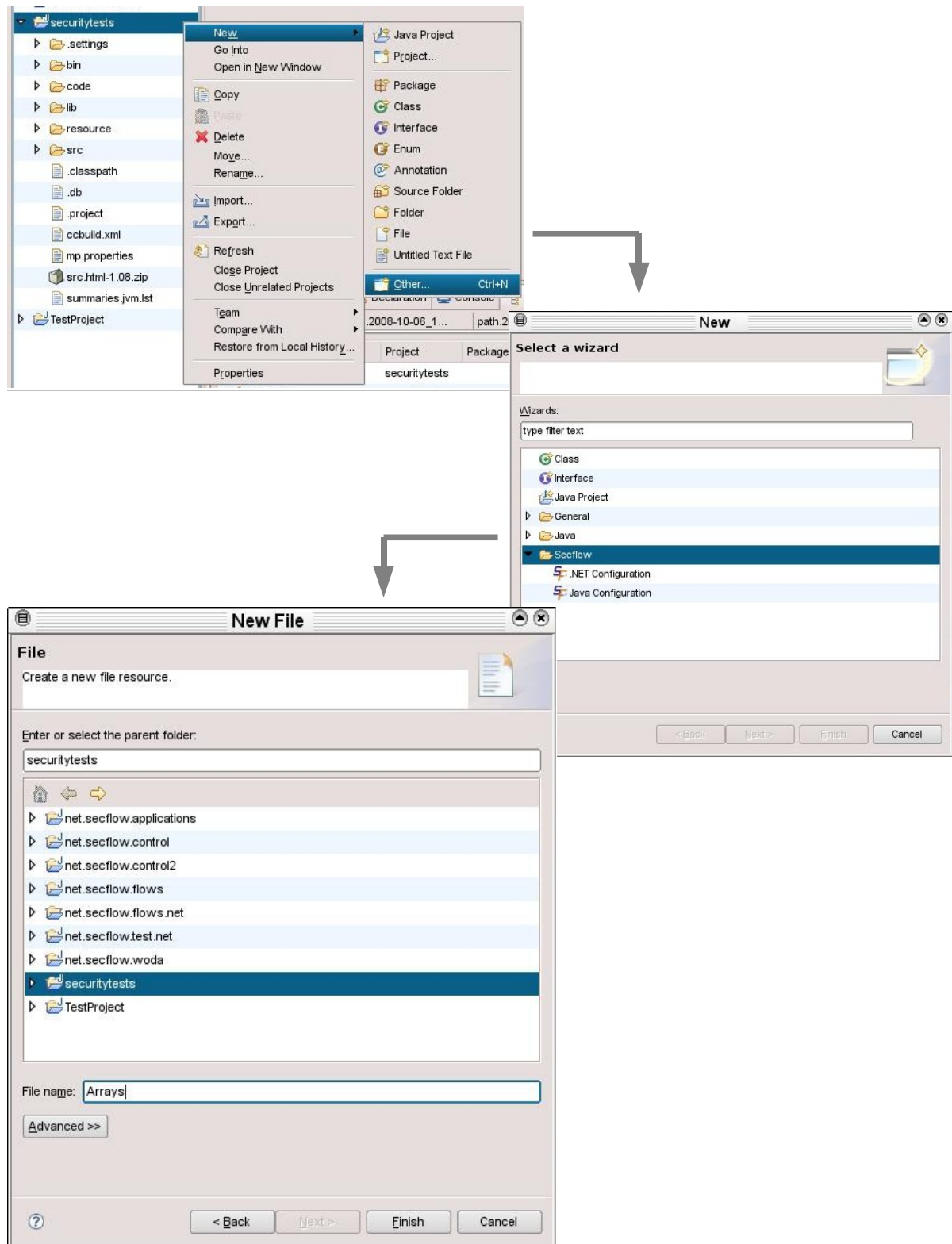


Abb.: Anlegen einer Konfigurationsdatei

Die angelegte Datei zeigt sich danach im Package- oder Navigation-Explorer und ist durch das SecFlow-Symbol  gekennzeichnet. Die Datei ist mit dem SecFlow Project Editor verknüpft. Dieser kann durch Doppelklick auf die Dateibezeichnung oder über das Kontextmenü, Menüpunkt "Open With|SecFlow Project Editor" geöffnet werden.

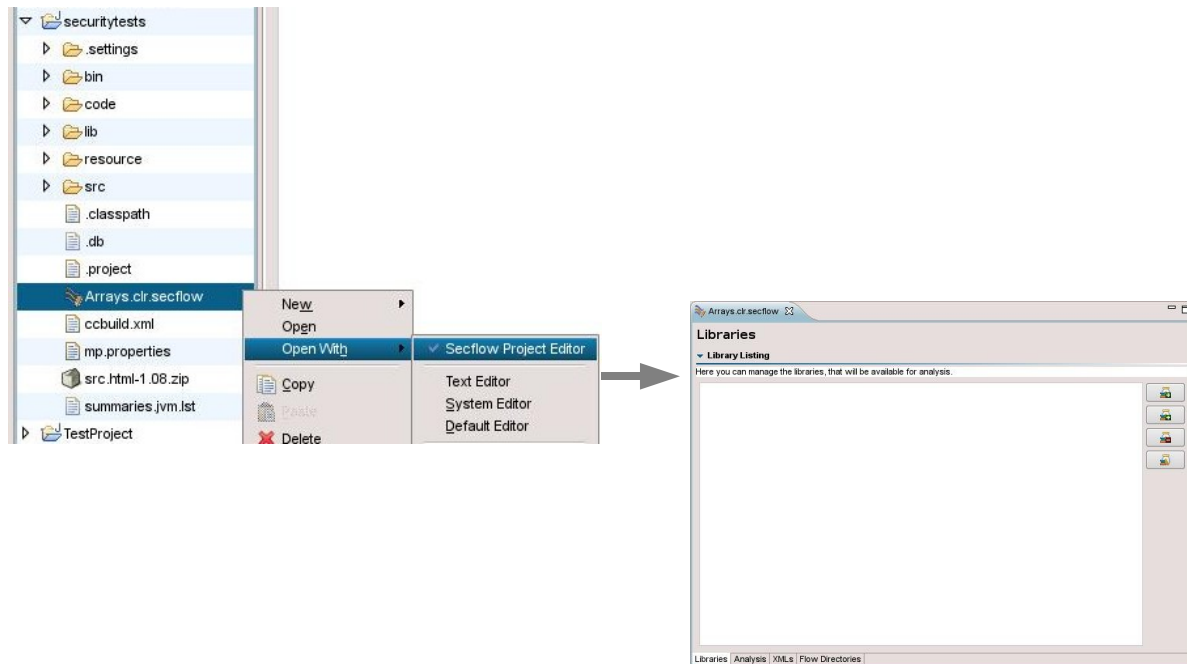


Abb.: Öffnen des SecFlow Projekt Editors

Bearbeiten einer Konfigurationsdatei

Mit dem SecFlow Project Editor ist es möglich eine Konfigurationsdatei zu editieren. Im SecFlow Project Editor existieren 3 (im Fall von Java) bzw. 4 Panels (im Fall von .NET).

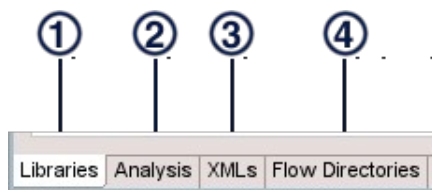


Abb.: Panels des SecFlow Projekt Editors

	Panel	Umgebung	Beschreibung
1	Libraries	.NET	Die zu analysierenden Klassen und nötige Systembibliotheken
		Java	Nur die zu analysierenden Klassen
2	Analysis	.NET	Entry-Methoden für die Analyse
		Java	Entry-Methoden für die Analyse
3	XMLs	.NET	XML-Dateien mit Debuginformationen
		Java	Panel nicht vorhanden
4	Flow Directories	.NET	Ordner mit benötigten Flow-Dateien
		Java	Ordner mit benötigten Flow-Dateien

4.4.2 SecFlow Project Editor

Der SecFlow Project Editor erlaubt es die Analyse genauer zu spezifizieren.

Libraries Panel

Das Libraries Panel zeigt die für die Analyse nötigen .NET-Assemblies. Unter .NET sind im Gegensatz zu Java die passenden .NET-Standard-Assemblies hinzuzufügen.

Die Checkboxes links neben den Assemblybezeichnern erlauben es, ein Assembly zur Analyse hinzuzufügen. Die im Assembly vorhandenen Klassen und Methoden werden darüber hinaus auf dem Analysis Panel auswählbar. Im Beispiel wurde darauf verzichtet die Systembibliotheken auszuwählen, da sie im allgemeinen durch Flow-Dateien abgebildet werden. Auf dem Analysis Panel sind sie somit nicht sichtbar.

Die Buttons auf der rechten Seite ermöglichen es, die Liste zu bearbeiten.

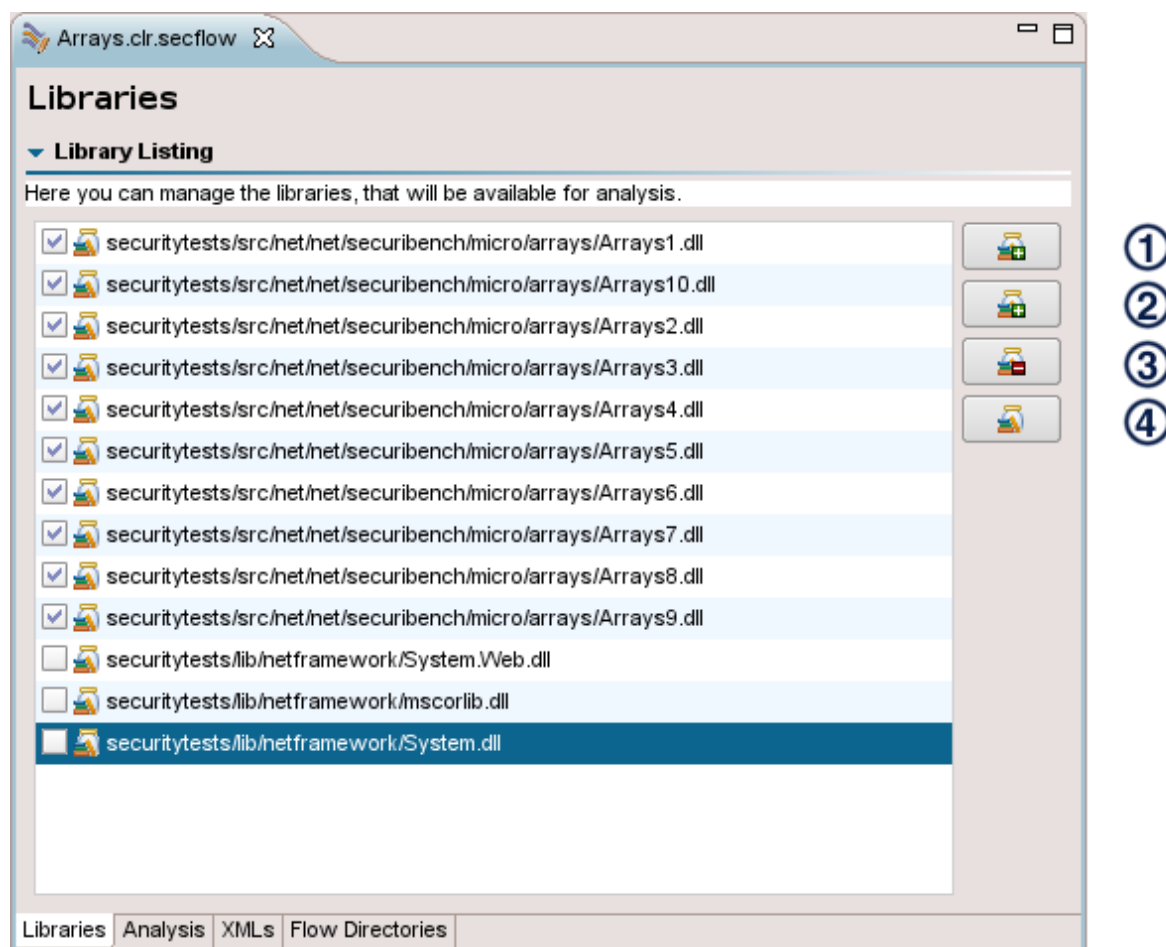


Abb.: Libraries Panel

Button	Beschreibung
1	Bibliothek aus dem Workspace hinzufügen
2	Externe Bibliothek hinzufügen
3	Bibliothek entfernen
4	Fehlende Bibliotheken anzeigen. Die Funktion dieses Buttons bezieht sich auf die per Checkbox ausgewählten Assembly-Dateien

Analysis Panel

Datei SecFlow_Benutzerhandbuch_V10.odt	Version 1.0	Datum 2008-11-27	Bearbeiter/in JK/DR	Seite 25 von 67
---	----------------	---------------------	------------------------	--------------------

Das Panel Analysis ermöglicht es Entry-Methoden auszuwählen, die als Startpunkte für die Suche nach kritischen Datenflüssen dienen. Die angezeigten Klassen und Methoden sind über die Auswahl auf dem Libraries Panel festgelegt.

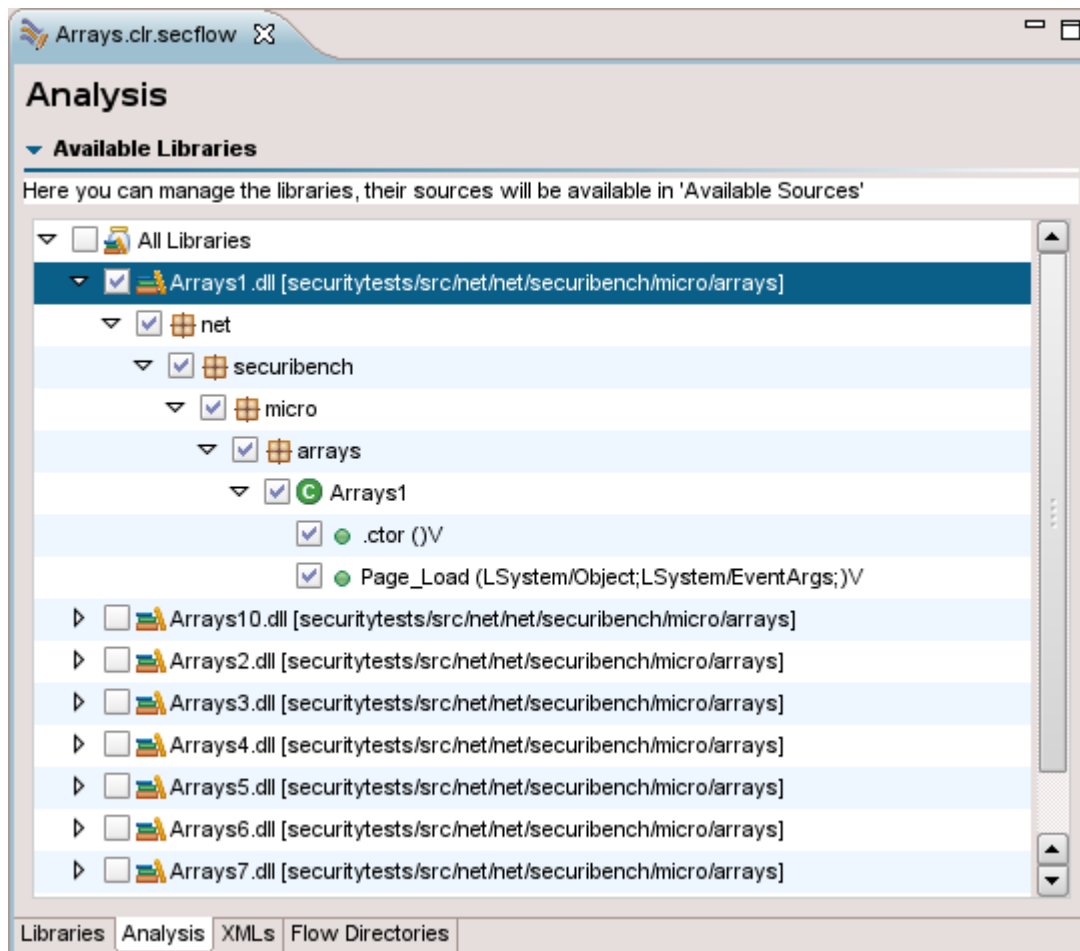


Abb.: Anaysis Panel

XMLs Panel

Unter Java sind Debuginformationen wie Sourcefile- oder Zeilenzuordnung in der Klassendatei enthalten. Unter .NET sind diese Informationen nicht im Assembly enthalten. Aussagekräftige Debuginformationen finden sich nur in den pdb-Dateien, die von Microsoft .NET Compilern erzeugt werden. Diese enthalten ähnliche Debuginformationen wie die Java Klassendateien. Die pdb-Dateien werden mit dem Windows-Tool pdb2xml in ein vom Framework lesbares XML-Format gebracht. Diese XML-Dateien können über das Panel ausgewählt werden.

Die Buttons auf der rechten Seite ermöglichen es die Liste zu bearbeiten.

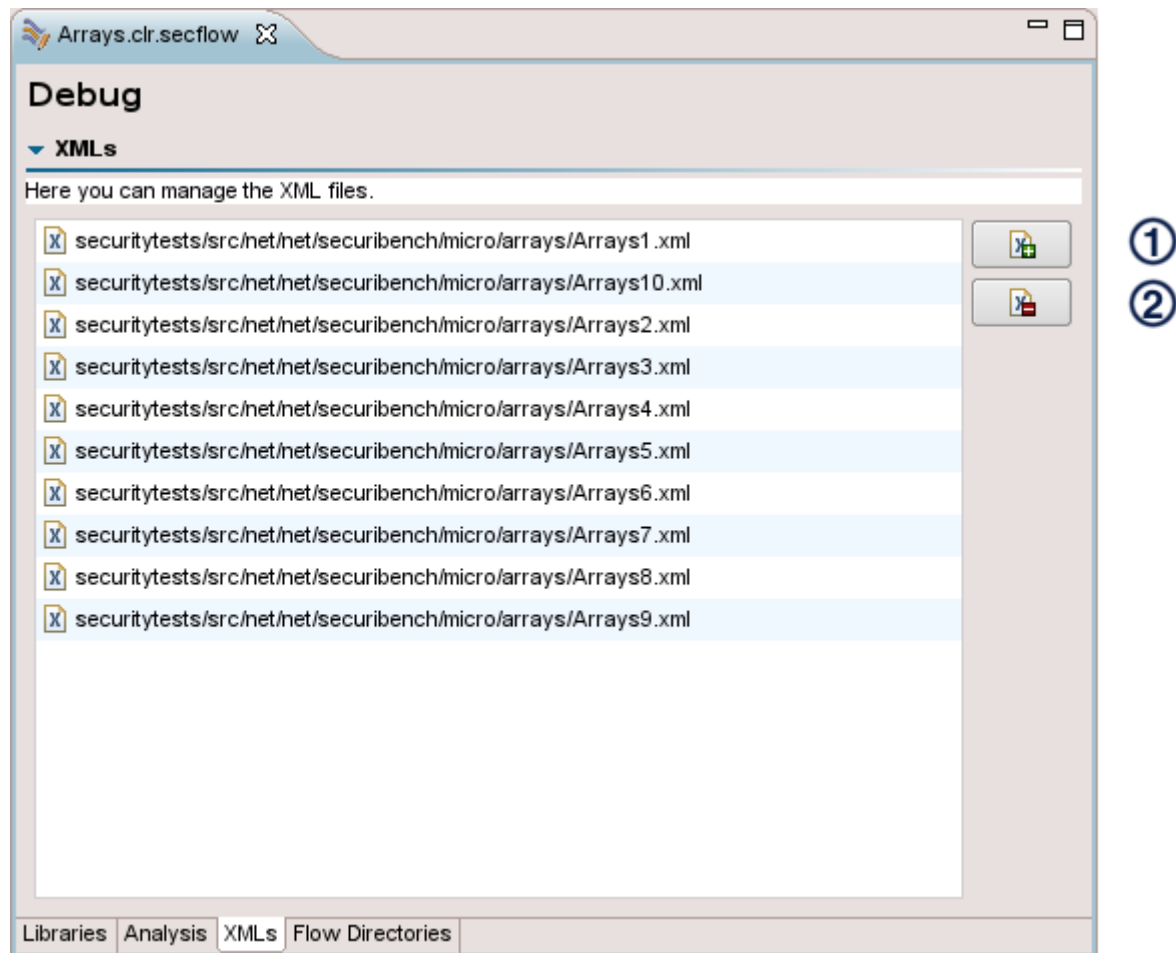


Abb.: XMLs Panel

Button	Beschreibung
1	Debugdatei aus dem Workspace hinzufügen
2	Debugdatei entfernen

Flow Directories Panel

Das Panel Flow Directories erlaubt die Auswahl von Ordnern, die Flowdateien enthalten. Der Inhalt der ausgewählten Ordner wird in der Gesamtübersicht der Flow View sichtbar.

Die Buttons auf der rechten Seite ermöglichen es die Liste zu bearbeiten.

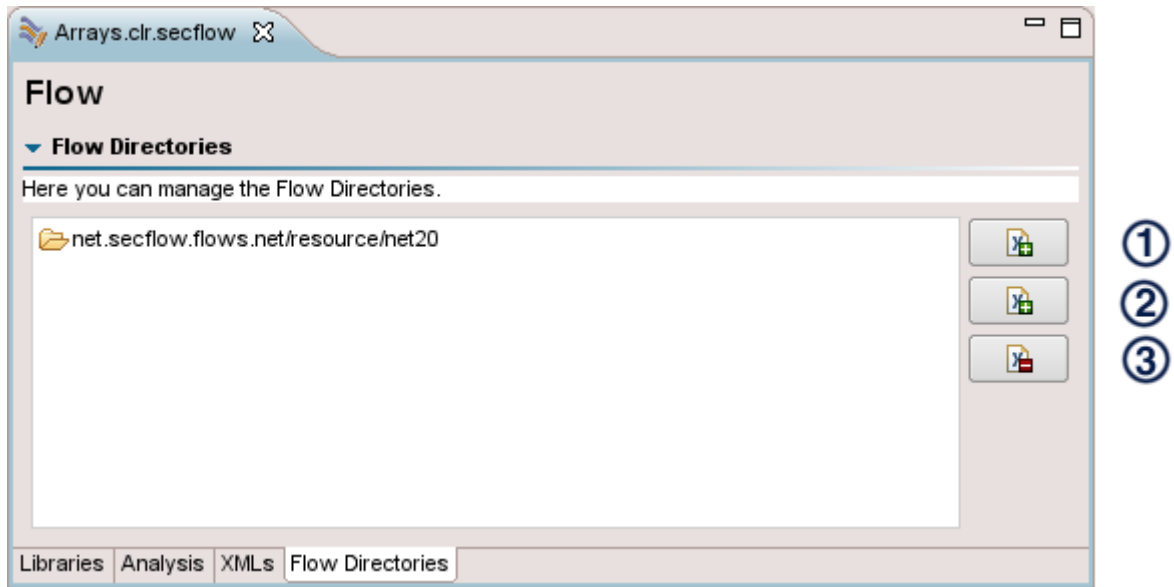


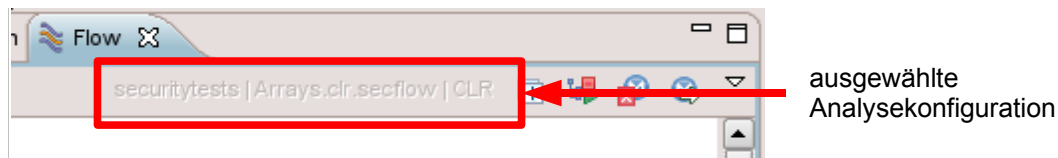
Abb.: Flow Directories Panel

Button	Beschreibung
1	Verzeichnis Flow-Dateien aus dem Workspace hinzufügen
2	Externes Verzeichnis für Flow-Dateien hinzufügen
3	Verzeichnis entfernen

4.4.3 Ausführung

In der Buttonleiste der Flow View wird die aktuell ausgewählte Analysekonfiguration angezeigt. Der Ausgabe hat folgende Struktur:

<Projekt><Konfigurationsdatei><Analyseumgebung>.



Die Analyse wird mit dem Button "Run SecFlow Analysis" im Eclipse Toolbar gestartet.



4.5 Views

Eclipse Views erlauben es, Informationen geordnet anzuzeigen und ihnen Operationen zuzuweisen. Die Views sollen bei der Konfiguration, Vorbereitung, Ausführung und Ergebnisanzeige unterstützen.

4.5.1 Library View

Der Library View zeigt die für ein Projekt verfügbaren Bibliotheksklassen (externe Funktionen) an.



Abb.: Librarysicht für ein CLR-Projekt



Abb.: Librarysicht für ein JVM-Projekt

Durch einen Rechtsklick auf einen oder mehrere markierte Einträge in der Library View lassen sich zu den ausgewählten Einträgen Summary-Dateien generieren.



Abb.: Summary-Dateien generieren

4.5.2 Flow View

Die Flow View zeigt die für ein Projekt verfügbaren Summary Dateien an. Weiterhin bietet sie Funktionen, um fehlende und unvollständige Summary Dateien im aktuellen Projekt zu bestimmen.

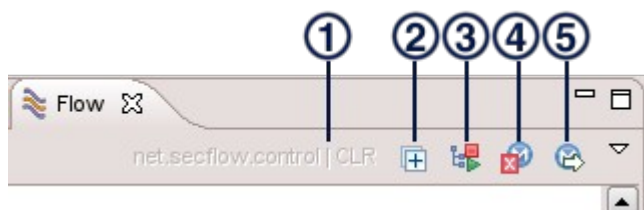


Abb.: Flow View Symbole

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	29 von 67

Die folgende Tabelle zeigt eine Übersicht mit den in der Flow View verfügbaren Funktionen:

Sektion	Beschreibung
1	Dieser Bereich der Flow View zeigt das aktuell für die Analyse aktive Projekt und die Analyseumgebung (JVM oder CLR). Der Bereich ist in die Kategorie Projektname (links) und Analyseumgebung (rechts) aufgegliedert (<Projektname> <Analyseumgebung>)
2	Dieser Knopf hat zwei Zustände:
	 Summary-Dateien zusammengeklappt, Objekteigenschaften ausgeblendet
	 Detailansicht der Summary-Dateien, Objekteigenschaften eingeblendet
3	Summary-Dateien anzeigen, die Sourcen und Senken enthalten
4	Alle für ein Projekt nötigen, aber fehlenden Summary-Dateien und Summary-Dateien mit fehlenden Methoden anzeigen
5	Alle für ein Projekt nötigen Summary-Dateien anzeigen

Das folgende Bild zeigt die zusammen gefaltete Darstellung der im Projektrahmen vorhandenen Summary-Dateien. Die einzelnen Bezeichner geben den Projektpfad und den Dateinamen wieder. Beispiel: net.secflow.flows/resource/net20/System.Char.flow bezeichnet die Summary-Datei System.Char.flow, die sich im Unterordner resource/net20 des Projektes net.secflow.flows befindet.

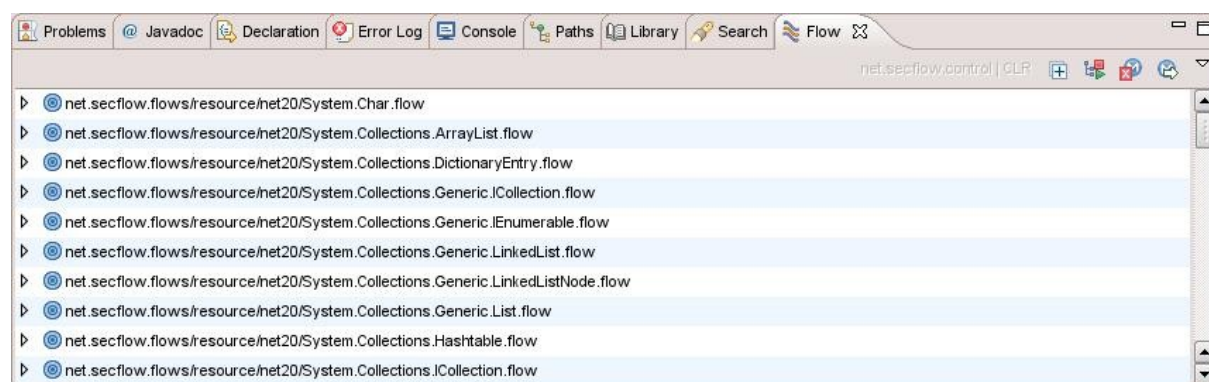


Abb.: Gefaltete Ansicht der Flow-Dateien

Durch einen Rechtsklick auf einen oder mehrere markierte Einträge in der Flow View lassen sich zu den ausgewählten Einträgen Summary-Dateien generieren.

Expandierte Ansicht der Flow-Dateien mit Objekteigenschaften:

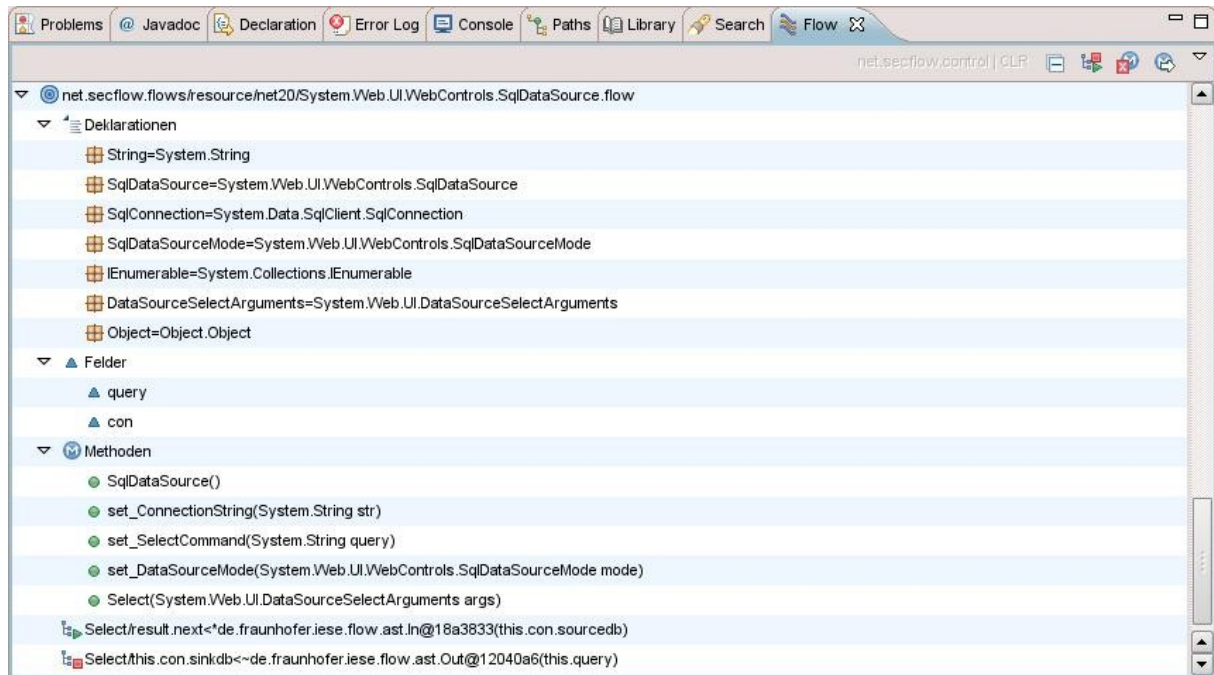


Abb.: Expandierte Ansicht der Flow-Dateien

In der View finden sich folgende Icons:

Icon	Bezeichnung
	Aktueller Eintrag ist der Bezeichner für eine Summary-Datei
	Importliste
	Import
	Feld bzw. Felderliste
	Methodenliste
	Methode
	Source (Quelle)
	Sink (Senke)

4.5.3 Flow Outline View

Die Flow Outline View zeigt eine strukturierte Übersicht über eine Flowdatei. Die View soll den Entwickler dabei unterstützen korrekte Flowdateien zu erstellen. Die Auswahl eines Elements öffnet im Sourceeditor die entsprechende Datei und markiert den betroffenen Sourcecode. Änderungen an der Flow Datei werden direkt in der Outline View übernommen.

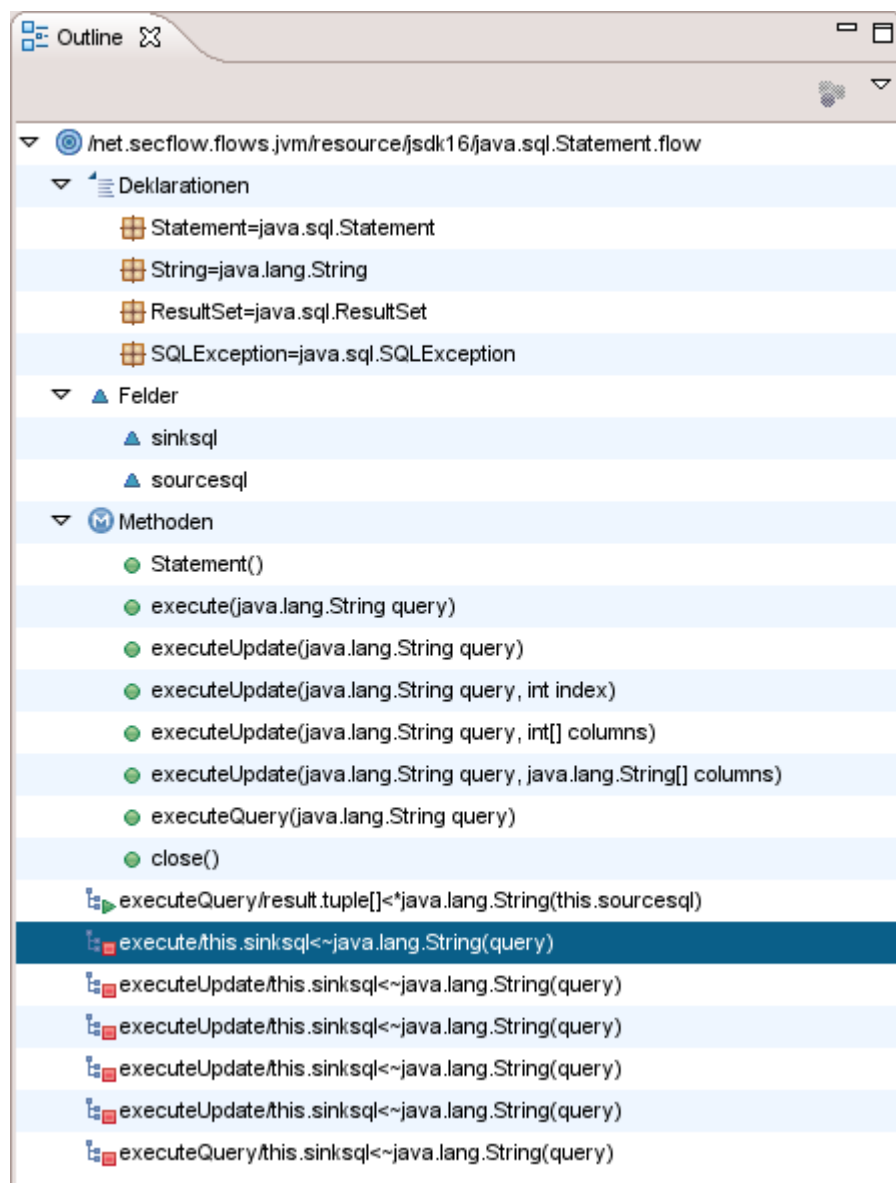


Abb.: Outline View

Die in der folgenden Tabelle aufgelisteten Icons beschreiben die verschiedenen Strukturelemente der ausgewählten Flowdatei.

Icon	Bezeichnung
	Verzeichnispfad der Flowdatei
	Deklarationen
	Deklaration mit Synonym und Langbezeichnung
	Feld bzw. Felderliste
	Methodenliste
	Methode mit Bezeichner und Signatur
	Source (Quelle)
	Sink (Senke)

HINWEIS: Wird eine Summary Datei nicht im Flow Outline View angezeigt, dann ist sie fehlerhaft erstellt.

4.5.4 Result View

Die Result View zeigt grundlegende Informationen zu einem Analyselauf:

- ◆ Betriebssystem Linux 32/64Bit | Windows 32/64Bit
- ◆ Verwendete Hardware CPU | Hauptspeicher
- ◆ Analysestatus Speicherverbrauch | Dauer | Ergebnis
- ◆ Pfadangaben
- ◆ .NET, Java Version
- ◆ BDD Konfiguration
- ◆ Kalkulierte Analysedauer

Optional können Ausgaben der Relationen angezeigt werden.

4.5.5 Path View

Die Path View enthält eine detaillierte Übersicht der Ergebnispfade, d.h. der verwundbaren Pfade. Die Inhalte ergeben sich aus den generierten out-Dateien in Verbindung mit den entsprechenden Sourcen.

Ein Click auf ein angezeigtes Pfadelement (Stellen, die Daten auf dem Weg von der Quelle zur Senke durchlaufen) öffnet den Sourceeditor und setzt eine Markierung im entsprechenden Quelltext. Somit ist eine Traversierung der angezeigten Pfadelemente im Quelltext möglich. Dies ermöglicht es dem Entwickler, relativ einfach Sanitizer zur Absicherung des Datenflusses an den entsprechenden Elementen in den Sourcecode einzubringen

Über die Path View lässt sich die Reportgenerierung ansteuern. Die in der Panelübersicht angezeigten Buttons erlauben die Reportausgabe in verschiedenen Formaten (siehe Kap. 5).

path.2008-10-04_22-21-17.out  path.out

Variable Id	Project	Package	File	Line	Expression
 Path #1	securitytests				
 3005551958287680	securitytests	net.securibench.micro.basic	Basic21.aspx.cs	24	Lnet/securibenc
 3005551958305608	securitytests	net.securibench.micro.basic	Basic21.aspx.cs	24	name
 3005551958305608	securitytests	net.securibench.micro.basic	Basic21.aspx.cs	38	name
 3005551958410520	securitytests	net.securibench.micro.basic	Basic21.aspx.cs	38	System.String.C
 Path #2	securitytests				
 3003118137240728	securitytests	net.securibench.micro.basic	Basic35.aspx.cs	35	Lnet/securibenc
 Path #3	securitytests				
 3000964331687560	securitytests	net.securibench.micro.session	Session2.aspx.cs	23	Lnet/securibenc
 3000964331726072	securitytests	net.securibench.micro.session	Session2.aspx.cs	23	name
 3000964331726072	securitytests	net.securibench.micro.session	Session2.aspx.cs	29	name
 3004866869252600	securitytests	net.securibench.micro.session	Session1.aspx.cs	32	Session.get_Itter
 3004866869253264	securitytests	net.securibench.micro.session	Session1.aspx.cs	32	(System.String)

Abb.: Path View (Ausschnitt)

Die in der folgenden Tabelle aufgelisteten Icons beschreiben die verschiedenen Pfadelemente.

Icon	Bezeichnung
	Pfad
	Source (Quelle)
	Element
	Sink (Senke)

Die einzelnen Spalten der View beinhalten folgende Angaben:

Spalte	Beschreibung
Variable Id	Element Identifier
Project	Eclipse Projektbezeichner
Package	Namensraum der betroffenen Klasse
File	Name der Sourcecodedatei
Line	Zeilennummer in der Sourcecodedatei
Expression	Betroffener Programmausdruck

4.6 Reporting

Die Reportgenerierung in SecFlow basiert auf der Reportingengine JasperReports und iReport (www.jasperforge.org). Das Tool iReport bietet eine GUI-Oberfläche zur Erstellung und Bearbeitung von JasperReports. Das Subreport-Konzept von iReport gestattet es, einzelne Komponenten eines Reports und den Report selbst in anderen Reports wiederzuverwenden. Dadurch ist es möglich, das Aussehen und die Ausgaben des Reportings ohne großen Aufwand zu ändern und zu erweitern.

Die Reportausgabe wird über die Buttons in der Path View (siehe Kap. 4.5.5) angesteuert. Die Reports zeigen die Analyseergebnisse in aufbereiteter Form und dienen dazu, die Analyseergebnisse in kompakter Form darzustellen, so dass sie nicht nur für den Entwickler sondern auch für andere Adressaten verständlich sind.

Neben der Gesamtübersicht ist es je nach Ausgabeformat auch möglich, den Pfad mittels Linkfelder in den ausgegebenen Sourcen zu traversieren. Die farbliche Darstellung wurde so gewählt, dass es möglich ist, die Pfade auf einfache Art und Weise nachzuvollziehen.

Die Gesamtübersicht im Report ist in Sektionen unterteilt:

Sektion	Beschreibung	
1	Neben der Überschrift findet man hier die analysierten Quellen und die Angabe, aus welchem Quellpaket diese stammen.	
2	Die Metrikübersicht beinhaltet statische Informationen bezüglich der Testergebnisse. Anhand vorkonfigurierter Schwellenwerte für OK, I (Informationen), W (Warning), E (Error) und S (Severe) werden die kumulierten Analyseergebnisse bewertet. Die Bewertung ist in der Spalte <i>Metric Grade</i> wiedergegeben. Zur Zeit werden folgende Metriken unterstützt:	
	1002 Number of Vulnerable Paths	Anzahl der gefundenen verwundbaren Pfade
	1003 Number of Sources	Anzahl der verschiedenen Pfadquellen
	1004 Number of Sinks	Anzahl der verschiedenen Pfadsenken
	1005 Number of Sanitizer	Anzahl der verschiedenen Pfadprotektoren (Operationen, die die Verlässlichkeit von Daten herstellen)
	1010 Maximum Path Length	Anzahl der Pfadelemente des längsten Pfades
	1015 Average Path Length	Durchschnittliche Anzahl der Pfadelemente
	1020 Affected Lines of Code	Anzahl der betroffenen Quelltextzeilen
3	Detaillierte Darstellung der gefundenen Pfade mit Identifier, Länge des Pfads, Quelltextzeilennummer des Pfadanfang, Quelltextzeilennummer des Pfadendes.	
4	Detaillierte Darstellung der gefundenen Pfadequellen mit Identifier, zugehörigem Pfad-Identifier, Quelltextzeilennummer des Auftretens und Ausdruck. Der Identifier (Source-ID) setzt sich aus einer 10stelligen Zahl, der korrespondierenden Pfad-ID und der Stelle des Elementes im Pfad zusammen.	

5	Detaillierte Darstellung der gefundenen Pfadesenken mit Identifier, zugehörigem Pfad-Identifier, Quelltextzeilennummer des Auftretens und Ausdruck. Der Identifier (Sink-ID) setzt sich aus einer 10stelligen Zahl, der korrespondierenden Pfad-ID und der Stelle des Elementes im Pfad zusammen.
6	Detaillierte Darstellung der gefundenen Pfadebereiniger mit Identifier, zugehörigem Pfad-Identifier, Quelltextzeilennummer des Auftretens und Ausdruck. Der Identifier (Sanitizer-ID) setzt sich aus einer 10stelligen Zahl, der korrespondierenden Pfad-ID und der Stelle des Elementes im Pfad zusammen.

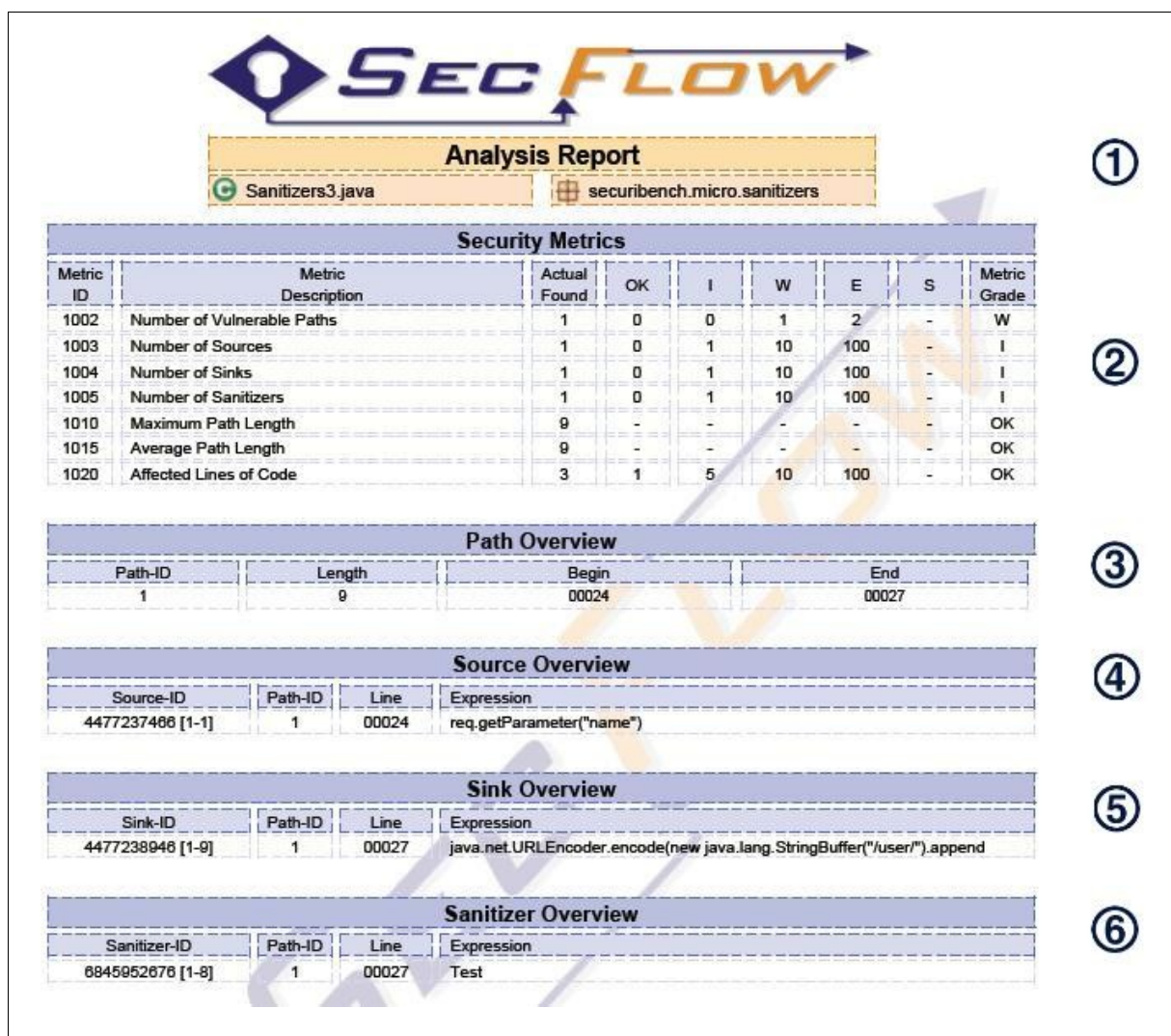


Abb.: Report - Gesamtübersicht

Die angezeigten Elemente aus 3 bis 6 sind über Hyperlinks mit der Detailsektion - der betroffenen Quelltextstelle – verbunden.

Im Detailbereich des Reports werden die analysierten Quelltexte abgebildet. Entsprechend der gefundenen Pfade, Quellen, Elemente und Senken werden weitere Anmerkungen hinzugefügt. In Rportausgaben in HTML- bzw. PDF-Format sind die Verbindungen zwischen den einzelnen Elementen über Querverweise dynamisiert.

```

00013 import securibench.micro.basicTestCase;
00014 import securibench.micro.MicroTestCase;
00015
00016 /**
00017  * @servlet description="safe redirect"
00018  * @servlet vuln_count = "0"
00019  */
00020 public class Sanitizers3 extends BasicTestCase implements MicroTestCase {
00021     private static final String FIELD_NAME = "name";
00022
00023     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
00024         String s = req.getParameter(FIELD_NAME);
00025
00026         String name = s.toLowerCase(Locale.UK);
00027         resp.sendRedirect(URLEncoder.encode("/user/" + name, "UTF-8")); /* OK */
00028     }
00029
00030     public String getDescription() {
00031         return "safe redirect";
00032     }
00033
00034     public int getVulnerabilityCount() {
00035         return 0;
00036     }

```

Source-ID	Path-ID	Expression
4477237466 [1-1]	1	req.getParameter("name")

Element-ID	Path-ID	Expression
4477237540 [1-2]	1	s

Element-ID	Path-ID	Expression
4477237540 [1-3]	1	s
4477237836 [1-4]	1	s.toLowerCase(java.util.Locale.UK)
4477237910 [1-5]	1	name

Element-ID	Path-ID	Expression
4477237910 [1-6]	1	name
4477238502 [1-7]	1	new java.lang.StringBuffer("/user/").append(name).toString()
6845952676 [1-8]	1	Test

Sanitizer-ID	Path-ID	Expression
6845952676 [1-8]	1	Test

Sink-ID	Path-ID	Expression
4477238946 [1-9]	1	java.net.URLEncoder.encode(new java.lang.StringBuffer("/user/").

Abb.: Report - Auszug Detailübersicht

5 Konfigurations-/Ausgabe-Dateien

5.1 Konfigurations-Datei *.secflow

Die Konfigurations-Datei erlaubt es Testfälle festzulegen. Die Dateien besitzen das Suffix secflow und sind im XML-Format abgelegt. Es existieren 3 Unterbereiche. Im ersten Bereich sind die für die Analyse benötigten Bibliotheken angegeben. Über den Schalter selected kann die Analyse und die Angabe von Klassen im Bereich classes weiter eingeschränkt werden. Der zweite Bereich nimmt die für die Zeilenrückverfolgung nötigen Debugdateien auf. Dieser Bereich ist in der aktuellen Version nur für die .NET Analyse interessant, da die Assemblydateien die benötigten Debug- und Zeileninformationen nicht enthalten. Im Gegensatz dazu enthalten die Java-Klassendateien bereits die nötigen Informationen. Der dritte Bereich enthält Klassen und Methoden die vom Framework analysiert werden sollen.

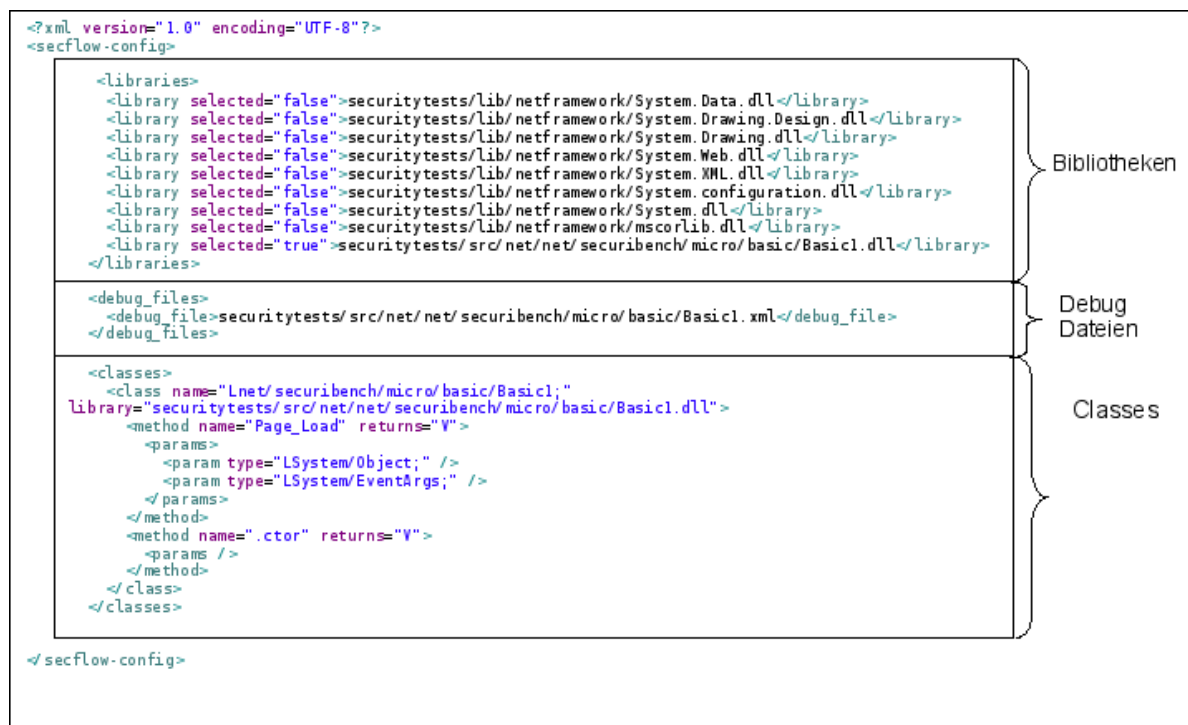


Abb.: Konfigurationsdatei *.secflow

Der strukturelle Aufbau der Datei ist:

- ◆ Das Wurzelement der Datei ist secflow-config
- ◆ libraries kann 0 bis n library Elemente einschließen.
- ◆ classes kann 0 bis n class Elemente einschließen
- ◆ class kann 0 bis n method Elemente einschließen
- ◆ flow_directories kann 0 bis n flow_directory Elemente einschließen

5.2 Flow-Dateien *.flow

Da jedes Quellprogramm auch Bibliotheken und andere Module nutzt, die zum Teil nicht im Quelltext vorliegen oder nicht analysiert werden sollen, muss dem Analyse-Werkzeug mitgeteilt werden, wo die Grenzen des Analyseumfangs liegen.

Gerade für die nicht zu analysierenden Teile muss dem Werkzeug bekannt gegeben werden, wie die Parameter des externen Modules zu bewerten sind. Dazu muss grob

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	38 von 67

skizziert werden, was mit den Eingabeparametern in der Methode passiert und wie das Ergebnis zu Stande kommt. Die Spezifizierung solcher Datenflusseigenschaften einer Methode werden Datenflussbeschreibung oder Flow-Spezifikation genannt. Sie wird in den Flow-Dateien vorgenommen.

ACHTUNG: Zur erfolgreichen Analyse eines Programms müssen **alle** nicht zu analysierenden Methoden in entsprechenden Flow-Dateien dargestellt werden.

5.2.1 Syntax einer FLOW-Datei

Grundstruktur

Die Grundstruktur einer flow-Datei ist wie folgt:

```
namedeclarations
```

```
summary <Name> {  
<attributes>  
<initializer/constructors/methods>  
}
```

→ Namensdeklaration

```
<Name> = '<Java-String>'
```

→ Summary

```
summary <Name> :  
<supertypes> {  
<attributes>  
<initializer/constructors/methods>  
}
```

→ Attribute

```
<AttributeName>;
```

→ Konstruktoren

Konstruktoren werden genauso notiert wie in Java oder C# und entsprechen auch deren Semantik:

```
<SummaryName>(<argumentlist>) {  
<code java>  
}
```

→ Methoden

Methoden werden genauso notiert wie in Java und entsprechen auch deren Semantik:

```
<Returntype> <Methodname>(<argumentlist>) {  
<code>  
}
```

→ Lokale Variablen

Lokale Variablen werden implizit bei der ersten Verwendung definiert.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	39 von 67

Zugriffe auf Attribute können nur über den vollständigen Objekt-Pfad, relativ zur umgebenden Aktion erfolgen:

```
result <- source.attribute // result ist lokale Variable
this.result <- source // result ist Attribut
```

→ Code

Code besteht aus Anweisungen, die durch Zeilenwechsel getrennt sind (also nicht etwa durch ein Semikolon):

```
statement1
statement2

Iterator iterator() {
result <- [Iterator]
result.element <- this.element
<- result
}
```

Quelle

→ an Attribut binden

Die Definition einer Quelle erfolgt durch eine Verknüpfung (~) eines Attributs mit einem Quellenausdruck (`Source('JavaString')`), z.B.

```
this.src ~ Source('HTTP')
```

Nach dieser Operation ist die HTTP-Quelle an das Attribut `this.src` gebunden. Bis auf weiteres definieren wir die möglichen Quellen-Ausdrücke noch nicht. Derzeit verwendet werden:

- HTTP: Quelle ist ein HTTP-Aufruf (Web-Formular, HTTP-Zeile)
- FILESYSTEM: Quelle ist das Dateisystem
- SQL: Quelle ist eine SQL-Datenbank

→ mit anderer Quelle verknüpfen

Die Verknüpfung einer Quelle eines Objekts mit der Quelle eines anderen Objekts erfolgt durch den Verknüpfungsoperator (~) zwischen den an die jeweilige Quelle gebundenen Attributen dieser Objekte, z.B.

```
this.src ~ source.src
```

Danach ist die Quelle, die ursprünglich nur an `source.src` gebunden war, nun auch an `this.src` gebunden.

→ lesen

Aus einer Quelle können Daten in Form eines Objekts gelesen werden (<*). Dabei steht die Quelle auf der Quellseite (rechts) als Argument eines funktionalen Ausdrucks, wobei der Name der Funktion identisch mit dem gelieferten Objekttyp ist. Das Datum steht auf der Zielseite (links).

```
data <* String(this.src)
```

data enthält ein Datum aus `this.src`.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	40 von 67

Senken

→ an Attribut binden

Die Definition einer Senke erfolgt durch eine Verknüpfung (~) eines Attributs mit einem Senkenausdruck (`Sink('JavaString')`), z.B.

```
this.snk ~ Sink('HTTP')
```

Nach dieser Operation ist `this.snk` an die HTTP-Senke gebunden.

Bis auf weiteres definieren wir die möglichen Senken-Ausdrücke noch nicht. Derzeit verwendet werden:

- HTTP: Senke ist eine HTTP-Adressangabe (Redirect, Forward)
- FILESYSTEM: Senke ist das Dateisystem SQL: Senke ist eine SQL-Datenbank
- REFLECTION: Senke ist eine reflektive Funktion
- SYSTEM: Senke ist ein Systemaufruf

→ verknüpfen

Die Verknüpfung einer Senke eines Objekts mit der Senke eines anderen Objekts erfolgt durch den Verknüpfungsoperator (~) zwischen den an die jeweilige Senke gebundenen Attributen dieser Objekte, z.B.

```
this.snk ~ sink.snk
```

Danach ist die Senke, die ursprünglich an `sink.snk` gebunden war, nun auch an `this.snk` gebunden.

→ schreiben

In eine Senke können Daten in Form eines Objekts geschrieben werden (<~). Dabei steht die Senke auf der Zielseite (links). Das Datum steht als Argument eines funktionalen Ausdrucks auf der Quellseite (rechts), wobei der Name der Funktion identisch mit dem erwarteten Objekttyp ist.

```
this.snk <~ String(data)
```

`data` fließt in die Senke.

Datenflüsse

→ Erhaltende Datenflüsse

Konservative Datenflüsse sind Datenflüsse, bei denen im Ziel des Flusses dasselbe Datum ankommt, das in der Quelle bereitgestellt wurde: Daten auf einem erhaltenden Fluss werden also nicht verändert. Erhaltende Datenflüsse haben folgende Syntax:

```
<target> <- <source>
```

Dabei dürfen beliebige Ausdrücke für `source` und `target` verwendet werden. Wenn das Ziel der Rückgabewert der Funktion ist, so muss `target` ausgelassen werden:

```
<- <source>
```

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	41 von 67

→ Generative Datenflüsse

Generative Datenflüsse sind Datenflüsse bei denen das Quelldatum eine Objekterzeugung ist. Sie haben folgende Syntax:

```
<target> <- [Type] // Erzeugung in einer Variablen  
<- [Type] // Erzeugung als Rückgabewert
```

→ Modifizierende Datenflüsse

Modifizierende Datenflüsse sind Datenflüsse, bei denen im Ziel des Flusses eine Modifikation der Quellen darstellt. Modifizierende Datenflüsse haben folgende Syntax:

```
<target> <- <modifiernamen>(<src>, <src>, ...) // zu einer Variablen  
<- <modifiernamen>(<src>, <src>, ...) // zum Rückgabewert
```

Derzeit sind folgende Modifier definiert:

sanitize	sanitized data cannot affect sinks of the sanitation type
validate	validated data must follow type conventions
enable	enables encoded parts of the data, revokes sanitize
split	result is the source splitted in parts
slice	result is a part of the source
replace	result is the source with replaced parts
chain	result is a combination of the sources
translate	result is a syntactical translation into another representation (e.g. ISO → UTF8)
todata	result is a semantic translation from a text into another representation ('9' → 9)
totext	result is a semantic translation from a representation into text(9 → '9')

Zur Analyse von Programmcode ist es nötig, möglichst viele Informationen über ein Programm in strukturierter Form vorliegen zu haben. Im SecFlow Framework werden diese Informationen in einer Deduktiven Datenbank gespeichert. Diese hat den Vorteil das auch rekursive Programmteile abgebildet werden können. Die logisch-relationale Programmiersprache Datalog wird dazu verwendet komplexe Muster aus Programmcode auszulesen.

Die Analyseregeln wurden in der logisch-relationalen Programmiersprache Datalog implementiert. Datalog ist die einfachste Logiksprache für relationale Datenbanken. Datalog ist der Sprache PROLOG sehr ähnlich. Über den Regelinterpreter (eine Weiterentwicklung des Werkzeuges BDDDB) werden die Regeln in einer Deduktiven Datenbank verwaltet. Das Datenbankschema wird durch die Files relations.datalog und rules.datalog dargestellt.

5.3 relation.datalog: Definition von Relationen

Die Relations-Datei enthält die grundlegenden strukturellen Informationen der Relationen ("Tabellendefinitionen"), d.h. die Spezifikation der gewünschten Ergebnistupel des Interpreters. Ein Relationsname (DB-Name) wird dabei als Prädikat mit Argumenten (Einträge der Datenbank) dargestellt.

Nachfolgend ist ein Auszug aus der Relations-Datei mit zusätzlichen Inline-Kommentaren abgebildet. Die vollständige inline-kommentierte Datei ist im SecFlow Sourcepaket enthalten und entsprechend verfügbar.

ACHTUNG: Die Datei ist nur im Zusammenhang mit Sourcecodeänderungen zu erweitern und zu ändern.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	42 von 67

```

# source(C, M, L, Inputregister, SrcRegister, NewType) iff at (C, M, L)
Inputregister is filled with a new piece of data of type NewType coming
from SrcRegister
#
# source("Ljavax/servlet/http/HttpServletRequest;", ".*", 8,
"this.parameterName", "this.http", "Ljava/lang/String;")
source(type:types, member:members, location:locations, register:registers,
src:registers, type:types) input

# sink(C, M, L, Register, SinkRegister, ConsumptionType) iff at (C, M, L)
Register flows to the sink SinkRegister being interpreted as type
Consumptiontype
#
# sink("Ljava/io/PrintWriter;", ".println (Ljava/lang/String;)V", 2,
"str", "this.print", "Ljava/lang/String;")
sink(type:types, member:members, location:locations, register:registers,
sink:registers, type:types) input

# transform (O, C, M, L, Object, Type) iff at (C, M, L) the operation O
initiates a Transformation of O to type Type
#
# transform("replace", "LSystem/Data/Common/DbCommand;", ".ExecuteNonQuery
()I", 6, "LSystem/Data/Common/DbCommand;.ExecuteNonQuery ()I@6", "~")
transform(operation:operations, type:types, member:members,
location:locations, object:objects, type:types) input

```

Abb.: Auszug aus der Datei "relation.datalog"

5.4 rules.datalog: Definition von Regeln

Mit der Regel-Datei lassen sich Ergebnistupel zur Analyse selbst definieren. Von ihnen ist primär die Qualität der Analyse und ihrer Ergebnisse abhängig.

Die Analyseregeln werden in Datalog wiedergegeben. Für eine Modifikation der Regeln sind daher erweiterte Kenntnisse dieser Programmiersprache unbedingt notwendig. Für weitere Informationen zu Datalog wird auf entsprechende Literatur verwiesen [1], [2].

Nachfolgend ist ein Auszug aus der Regel-Datei mit zusätzlichen Inline-Kommentaren abgebildet. Die vollständige inline-kommentierte Datei ist im SecFlow Sourcepaket enthalten und entsprechend verfügbar.

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	43 von 67

```

#class objects are implicitly generated
variablegen("", "@types", 0, Dest, "", Object, 0) :- \
    type2register(Type, Dest), \
    type2object(Type, Object).

# variable flows (meaning flows, where content is not changed)
variableflow(DestType, DestMember, DestContext, Dest, DestPath, SrcType,
SrcMember, SrcContext, Src, SrcPath) :- \
    cs_dependency(Operation, DestType, DestMember, DestLocation,
DestContext, Dest, DestPath, SrcType, SrcMember, _, SrcContext, Src,
SrcPath), \
    !transform(Operation, DestType, DestMember, DestLocation, *, *).

# base case: a variable points to some location if it is defined in a
variable gen
variablepointsto(Type, Member, ObjectContext, ObjectRegister, Object,
ObjectId) :- \
    variablegen(Type, Member, ObjectContext, ObjectRegister, "", Object,
ObjectId).
# induction step: A variable points to an object if there is a flow from a
variable pointing to this object
variablepointsto(Type, Member, ObjectContext, ObjectRegister, Object,
ObjectId) :- \
    variableflow(Type, Member, ObjectContext, ObjectRegister, "", ValueType,
ValueMember, ValueContext, ValueRegister, ""), \
    variablepointsto(ValueType, ValueMember, ValueContext, ValueRegister,
Object, ObjectId).

```

Abb.: Auszug aus der Datei "rules.datalog"

5.5 Pfad-Datei *.out

Der Test generiert eine Datei mit dem Suffix out. Diese beinhaltet in XML Form die gefundenen verwundbaren Pfade und deren Subelemente (Quellen, Elemente, Senken).

<pre> <paths count="1"> <path id="1" rank="834"> </pre>		
<pre> <variable id="5276605140"> <file>securibench/micro/datastructures/Datastructures8.java</file> <scope>Lsecuribench/micro/datastructures/Datastructures8;.doGet (...)V</scope> <line>35</line> <expression>req.getParameter("&name&quot;)</expression> <attribute key='tooltip' value='always validate the result'/> <threat>HTTP</threat> </variable> </pre>	Quelle	
<pre> <operation name="MOVE_A" from="5276605140" to="5276605230"/> </pre>	Operation	
<pre> <variable id="5276605230"> <file>securibench/micro/datastructures/Datastructures8.java</file> <scope>Lsecuribench/micro/datastructures/Datastructures8;.doGet (...)V</scope> <line>35</line> <expression>name</expression> </variable> </pre>	Element	
<pre> <variable id="5276606040"> <file>securibench/micro/datastructures/Datastructures8.java</file> <scope>Lsecuribench/micro/datastructures/Datastructures8;.doGet (...)V</scope> <line>46</line> <expression>str</expression> <threat>HTML</threat> </variable> </pre>	Senke	
<pre> </path> </paths> </pre>		

Abb.: Beispiel einer Pfad-Datei

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	44 von 67

Der strukturelle Aufbau der Datei ist:

- Das Wurzelement der Datei ist `paths`.
 - Das Attribut `count` gibt die Anzahl der `path` Tochterelemente an.
- `paths` kann 0 bis `n` `path` Elemente einschließen.
 - Das Element `id` Gibt die Anzahl der `path` Tochterelemente an.
- `path` kann 1 bis `n` `variable` Elemente und 0 bis `n` `operation` Elemente beinhalten.
- Das `variable` Element besteht aus der `id` und weiteren Subelementen.
 - Das Attribut `id` ist ein eindeutiger Bezeichner, der verschiedene Variablen voneinander trennt.
 - Das Subelement `File` nennt den Pfad der Quelldatei, in dem sich das Element befindet.
 - Das Subelement `Scope` zeigt auf die Stelle innerhalb der Quelldatei, an welcher sich das Element befindet.
 - Das Subelement `Line` zeigt auf die Zeile innerhalb der Quelldatei, an welcher sich das Element befindet.
 - Das Subelement `Expression` zeigt auf den Ausdruck innerhalb der Quelltextzeile, an welcher sich das Element befindet.
 - Das Subelement `attribute` beinhaltet den Verweis zur Wissensbasis bezüglich des Elements. Dieses Subelement gibt es nur bei Quellen.
 - Das Subelement `threat` bezeichnet den Gefahrentyp des Elementes. Dieses Subelement gibt es nur bei Quellen und Senken.
- Das zusätzliche Element `operation` beschreibt die Art des Flusses (`name`), sowie die Flussrichtung (`from/to`) der Daten zwischen Variablen. Für die detaillierte Darstellung der verschiedenen Attributwerte von `name` wird auf die Seiten der Stanford Universität¹ (<http://suif.stanford.edu/~jwhaley/>) verwiesen.

5.6 Quad und seine Komponenten

Zur internen Programmdarstellung werden Kontrollflussgraphen eingesetzt. Sie bestehen aus Basic Blöcken einfacher Befehlsrepräsentationen, genannt Quads. Quads bestehen aus einem Operator und bis zu vier Operanden. Für eine tiefergehende Betrachtung von Quad verweisen wir auf das Dokument <http://cycleserv2.csail.mit.edu/Harpoon/quads/quads.pdf>, sowie auf http://suif.stanford.edu/~courses/cs243/joeq/quad_overview.html.

5.6.1 Operatoren in Quad

Operator	Kurz- Beschreibung	Unterklassen	Methoden
Move	Wert zwischen zwei Pseudo Registern verschieben	MOVE_{I,F,L,D,A}	getMoveOp, getDest, getSrc
Binary	Binäre Anweisung, mit 2 Quellen und einem Ziel	ADD_{I,L,F,D}, SUB_{I,L,F,D}, MUL_{I,L,F,D}, DIV_{I,L,F,D}, REM_{I,L,F,D}, AND_{I,L}, OR_{I,L}, XOR_{I,L}, SHL_{I,L}, SHR_{I,L}, USHR_{I,L}, CMP_{L,F,D}	getDest, getSrc1, getSrc2
Unary	Unäre Anweisung, mit einer Quelle und einem Ziel	NEG_{I,F,L,D}, INT_2LONG, INT_2FLOAT, INT_2DOUBLE, LONG_2INT, LONG_2FLOAT, LONG_2DOUBLE, FLOAT_2INT, DOUBLE_2LONG, DOUBLE_2FLOAT, INT_2BYTE, INT_2CHAR, INT_2SHORT, OBJECT_2INT, INT_2OBJECT, FLOAT_2INTBITS, INTBITS_2FLOAT, DOUBLE_2FLOAT, DOUBLE_2LONGBITS, LONGBITS_2DOUBLE	getDest, getSrc
Goto	Unbedingter Sprung	GOTO	getTarget
IntIfCmp	Bedingter Sprung	IFCMP_{I,A}	getSrc1, getSrc2, getCond, getTarget
TableSwitch	Table Switch (Index)	TABLESWITCH	setTarget, getSrc, getDefault, getLow, getTarget, getTargetTable
LookupSwitch	Lookup Switch (Key)	LOOKUPSWITCH	setMatch, setTarget, getSrc, getMatch, getTarget, getValueTable, getTargetTable
Getstatic	Lade von einem statischen Feld	GETSTATIC_{I,F,L,D,A}, GETSTATIC_{I,F,L,D,A}_ DYNLINK	getDest, getField

Operator	Kurz- Beschreibung	Unterklassen	Methoden
Putstatic	Speichere in einem statischen Feld	PUTSTATIC_{I,F,L,D,A}, PUTSTATIC_{I,F,L,D,A}_ DYNLINK	getSrc, getField
Getfield	Lade von einem Objekt Feld	GETFIELD_{I,F,L,D,A,B, C,S,Z}, GETFIELD_{I,F,L,D,A,B, C,S,Z}_DYNLINK	getDest, getBase, getField, getGuard
Putfield	Speichere in einem Objekt Feld	PUTFIELD_{I,F,L,D,A,B, C,S,Z}, PUTFIELD_{I,F,L,D,A,B, C,S,Z}_DYNLINK	getBase, getSrc, getField, getGuard
ALoad	Lade Wert aus Array	ALOAD_{I,L,F,D,A,B,C,S}	getDest, getIndex, getGuard
ASore	Speichere Wert in Array	ASTORE_{I,L,F,D,A,B,C, S}	getValue, getBase, getIndex, getGuard
ALength	Länge eines Arrays	ARRAYLENGTH	getDest, getSrc
BoundsCheck	Array Bereichs Prüfung	BOUNDS_CHECK	getRef, getIndex, getGuard
NullCheck	Nullpointer Prüfung	NULL_CHECK	getDest, getSrc
ZeroCheck	0 Prüfung	ZERO_CHECK	getDest, getSrc
StoreCheck		ASTORE_CHECK	getRef, getElement, getGuard
New	Object instanzieren	NEW	getDest, getType
NewArray	Array instanzieren	NEWARRAY	getDest, getSize, getType
CheckCast	Typprüfung	CHECK_CAST	getDest, getSrc, getType
InstanceOf	Instanzprüfung	INSTANCEOF	getDest, getSrc, getType
Invoke	Methodenaufruf	INVOKEVIRTUAL_{V,I,F, L,D,A}, INVOKESTATIC_{V,I,F,L, D,A}, INVOKEVIRTUAL_{V,I,F, L,D,A}_DYNLINK, INVOKESTATIC_{V,I,F,L, D,A}_DYNLINK, INVOKESPECIAL_{V,I,F, L,D,A}_DYNLINK, INVOKEINTERFACE_{V, I,F,L,D,A}	setParam, getDest, getMethod, getParam, getParamList
Jsr	Aufruf einer Subroutine	JSR	getDest, getTarget

Operator	Kurz- Beschreibung	Unterklassen	Methoden
Ret	Rücksprung von einer internen Subroutine	RET	getTarget
Return	Rücksprung von der aktuellen Methode	RETURN_{V,I,F,L,D,A}, THROW_A	getSrc
Monitor		MONITORENTER, MONITOREXIT	getSrc

5.6.2 Operanden in Quad

Operand	Beschreibung
RegisterOperand	Abstrakte Speicherstelle
IConstOperand	Integer Konstante (32 bit)
FConstOperand	Floating Point Konstante (32 bit)
LConstOperand	Long Integer Konstante (64 bit)
DConstOperand	Double-precision Floating Point Konstante (64 bit)
AConstOperand	Object Referenz
MethodOperand	Zielmethode eines Methodenaufrufs
ParamListOperand	Parameterliste für einen Methodenaufruf
FieldOperand	Objekt Feld verbunden mit get/put
TypeOperand	Typenoperande für Typenprüfbefehl
ConditionOperand	Bedingung in Verbindung mit einem Bedingten Sprungbefehl
TargetOperand	Ziel eines Sprungbefehls
BasicBlockTableOperand	Sprungzieltabelle für die Verwendung in Switch-Befehlen

6 Verwendungsbeispiele

6.1 SQL-Injection

Art der betroffenen Bibliotheken : Klasse

Beschreibung: Ein Servlet baut eine SQL-Query mit Hilfe eines ungeprüften Parameters auf. Der Parameter kann einen weiteren zusätzlichen SQL-Befehl enthalten, der von der Anwendung "unwissentlich" ausgeführt werden und damit ungewollte Reaktionen nachsichziehen kann.

Ausgangscode des Beispiels:

```
public class Appl extends HttpServlet {

    protected void doGet( HttpServletRequest request, HttpServletResponse
response )
    {

        String dbURL = "jdbc:mysql://localhost:3306/shopDb";
        String dbDriver = "com.mysql.jdbc.Driver";
        String dbUser   = "shop";
        String dbPwd    = "myShopPassword";
        String sqlQuery = "SELECT author, subject, text FROM tbBook WHERE
bookID=";

        String bookId = request.getParameter( "bookId"); /* Hier wird aus einer
                                                         unsicheren HTTP-
                                                         Quelle gelesen */

        sqlQuery = sqlQuery+bookId; /* SQL Befehl wird zusammengebaut */

        PrintWriter out = null;
        try { out = response.getWriter(); }
        catch (IOException e1) { e1.printStackTrace(); return; }

        out.println("<html><head><title>Book List</title></head><body>");

        Connection conn=null; Statement stat=null; ResultSet rs=null;

        try{
            Class.forName( dbDriver ); // load driver
            conn = DriverManager.getConnection( dbURL, dbUser, dbPwd);
            stat = conn.createStatement();

            rs    = stat.executeQuery( sqlQuery ); /* Hier wird in eine
                                                         kritische SQL-Senke
                                                         geschrieben */

            while( rs.next() ) {
                out.print("Author: " + rs.getString(1));
                out.print(" Subject: " + rs.getString(2));
                out.print(" Text: " + rs.getString(3));
                out.println("<br>");
            }
        }
        catch ( Exception e ){ e.printStackTrace(); }

        out.println("</body></html>");

    }

}
```

Flowfiles:

javax.servlet.http.HttpServletRequest.flow

Im Beispielcode wird aus einer unsicheren Http-Quelle gelesen. Da diese nicht durch SecFlow analysiert werden kann, muß sie durch eine Flow-Datei beschrieben werden. Die Flow-Datei sieht wie folgt aus:

```
summary HttpServletRequest : ServletRequest {

    http;
    parameter;
    parameterName;

    /* Initialisierung */
    {
        this.http ~ source('HTTP')          /* HTTP-Quelle Definition */
        this.parameter <* String(this.http)  /*Parameter liest aus
                                           HTTP-Quelle */

        this.parameterName <* String(this.http)
        ...
    }

    ...

    String getParameter(String parametername) {
        <- this.parameter          /* Datenfluss aus HTTP-Quelle */
    }
}
```

java.sql.Connection.flow

Diese Summary definiert die kritische Senke.

```
summary Connection {

    sinksql;
    sourcesql;

    {
        this.sinksql ~ sink('SQL')          /* Definition SQL-Senke*/
        this.sourcesql ~ source('SQL')
    }

    Statement createState() {
        result <- [Statement]
        result.sinksql ~ this.sinksql /* Die Quelle des hier erzeugten
                                     Statements ist eine SQL-Quelle */

        result.sourcesql ~ this.sourcesql
        <- result
    }
}
```

java.sql.Statement.flow

Folgende FLOW-Datei beschreibt die Datenflüsse in der Klasse Statement, u. A. das Schreiben in eine Senke (diese wurde in der java.sql. Connection.flow als SQL-Senke definiert).

```
summary Statement {  
  
    sinksql;  
    sourcesql;  
  
    ...  
  
    ResultSet executeQuery(String query) {  
this.sinksql <~ String(query)          /* schreiben in die Senke */  
        result <- [ResultSet]  
        result.tuple[] <~ String(this.sourcesql)  
        <- result  
        ...  
        sqlexception.msg <- query  
        ...  
    }  
}
```

Report:



Analysis Report

App1.java

ict.net.ictteam.cms.utils

Security Metrics

Metric ID	Metric Description	Actual Found	OK	I	W	E	S	Metric Grade
1002	Number of Vulnerable Paths	1	0	0	1	2	-	W
1003	Number of Sources	1	0	1	10	100	-	I
1004	Number of Sinks	1	0	1	10	100	-	I
1005	Number of Guards	0	0	1	10	100	-	OK
1010	Maximum Path Length	6	-	-	-	-	-	OK
1015	Average Path Length	6	-	-	-	-	-	OK
1020	Affected Lines of Code	3	1	5	10	100	-	OK

Path Overview

Path-ID	Length	Begin	End	Rank
1	6	00026	00044	644

Source Overview

Source-ID	Path-ID	Line	Expression
20454946914 [1-1]	1	00026	request.getParameter("bookId")

Sink Overview

Sink-ID	Path-ID	Line	Expression
20454948432 [1-6]	1	00044	sqlQuery

Guard Overview

Guard-ID	Path-ID	Line	Expression
----------	---------	------	------------

Sanitizer Overview

Sanitizer-ID	Path-ID	Line	Expression
--------------	---------	------	------------

Annotated Source Code

Source #1
App1.java
ict.net.ictteam.cms.utils

Line	Code												
00001	package ict.net.ictteam.cms.utils;												
00002													
00003	import java.io.IOException;												
00004	import java.io.PrintWriter;												
00005	import java.sql.Connection;												
00006	import java.sql.DriverManager;												
00007	import java.sql.ResultSet;												
00008	import java.sql.Statement;												
00009													
00010	import javax.servlet.http.HttpServlet;												
00011	import javax.servlet.http.HttpServletRequest;												
00012	import javax.servlet.http.HttpServletResponse;												
00013													
00014	public class App1 extends HttpServlet {												
00015													
00016	protected void doGet(HttpServletRequest request, HttpServletResponse response)												
00017	{												
00018													
00019	String dbURL = "jdbc:mysql://localhost:3306/shopDb";												
00020	String dbDriver = "com.mysql.jdbc.Driver";												
00021	String dbUser = "shop";												
00022	String dbPwd = "myShopPassword";												
00023	String sqlQuery = "SELECT author, subject, text FROM tbBook WHERE bookID=";												
00024													
00025	/* Hier wird aus unsicheren HTTP-Quelle gelesen */												
00026	String bookId = request.getParameter("bookId"); /* Zeile 29 */												
	<table border="1" style="width: 100%; border-collapse: collapse; background-color: #ffe0e0;"> <thead> <tr> <th>Source-ID</th> <th>Path-ID</th> <th>Expression</th> </tr> </thead> <tbody> <tr> <td>20464946014 [1-1]</td> <td>1</td> <td>request.getParameter("bookId")</td> </tr> </tbody> </table>	Source-ID	Path-ID	Expression	20464946014 [1-1]	1	request.getParameter("bookId")						
Source-ID	Path-ID	Expression											
20464946014 [1-1]	1	request.getParameter("bookId")											
	<table border="1" style="width: 100%; border-collapse: collapse; background-color: #ffffe0;"> <thead> <tr> <th>Element-ID</th> <th>Path-ID</th> <th>Expression</th> </tr> </thead> <tbody> <tr> <td>20464947052 [1-2]</td> <td>1</td> <td>bookId</td> </tr> </tbody> </table>	Element-ID	Path-ID	Expression	20464947052 [1-2]	1	bookId						
Element-ID	Path-ID	Expression											
20464947052 [1-2]	1	bookId											
00027													
00028	sqlQuery = sqlQuery+bookId; /* SQL Befehl wird zusammengebaut */												
	<table border="1" style="width: 100%; border-collapse: collapse; background-color: #ffffe0;"> <thead> <tr> <th>Element-ID</th> <th>Path-ID</th> <th>Expression</th> </tr> </thead> <tbody> <tr> <td>20464947052 [1-3]</td> <td>1</td> <td>bookId</td> </tr> <tr> <td>20464948294 [1-4]</td> <td>1</td> <td>new java.lang.StringBuilder(java.lang.String.valueOf("SELECT</td> </tr> <tr> <td>20464948432 [1-5]</td> <td>1</td> <td>sqlQuery</td> </tr> </tbody> </table>	Element-ID	Path-ID	Expression	20464947052 [1-3]	1	bookId	20464948294 [1-4]	1	new java.lang.StringBuilder(java.lang.String.valueOf("SELECT	20464948432 [1-5]	1	sqlQuery
Element-ID	Path-ID	Expression											
20464947052 [1-3]	1	bookId											
20464948294 [1-4]	1	new java.lang.StringBuilder(java.lang.String.valueOf("SELECT											
20464948432 [1-5]	1	sqlQuery											
00029													
00030	PrintWriter out = null;												
00031	try { out = response.getWriter(); }												
00032	catch (IOException e) { e.printStackTrace(); return; }												
00033													
00034	out.println("<html><head><title>Book List</title></head><body>");												
00035													
00036	Connection conn=null; Statement stat=null; ResultSet rs=null;												
00037													
00038	try{												
00039	Class.forName(dbDriver); // load driver												
00040	conn = DriverManager.getConnection(dbURL, dbUser, dbPwd);												
00041	stat = conn.createStatement();												
00042													
00043	/* Hier wird in kritische SQL Senke geschrieben */												
00044	rs = stat.executeQuery(sqlQuery); /* Zeile 47 */												
	<table border="1" style="width: 100%; border-collapse: collapse; background-color: #e0ffe0;"> <thead> <tr> <th>Sink-ID</th> <th>Path-ID</th> <th>Expression</th> </tr> </thead> <tbody> <tr> <td>20464948432 [1-6]</td> <td>1</td> <td>sqlQuery</td> </tr> </tbody> </table>	Sink-ID	Path-ID	Expression	20464948432 [1-6]	1	sqlQuery						
Sink-ID	Path-ID	Expression											
20464948432 [1-6]	1	sqlQuery											
00045													
00046	while(rs.next()) {												
00047	out.print("Author: " + rs.getString(1));												
00048	out.print(" Subject: " + rs.getString(2));												
00049	out.print(" Text: " + rs.getString(3));												
00050	out.println(" ");												
00051	}												
00052	catch (Exception e) { e.printStackTrace(); }												
00053													
00054	out.println("</body></html>");												
00055													
00056	}												
00057	}												

6.2 XSS – Cross Site Scripting

Art der betroffenen Bibliotheken :	Klasse
Beschreibung:	Das Servlet entnimmt den User-Namen aus einem ungeprüften http-Request-Parameter. Der Parameter kann schädlichen HTML-Code bzw. JavaScript-Code enthalten.

Ausgangscode des Beispiels:

```
public class App2 extends HttpServlet {  
  
    protected void doGet(    HttpServletRequest    request,  
        HttpServletResponse response )  
    {  
        String userName = request.getParameter( "userName");  
        /* Hier wird aus unsicheren HTTP-Quelle gelesen */  
  
        PrintWriter out = null;  
  
        try { out = response.getWriter();} /* ein Writer mit  
                                           HTML-Senke */  
        catch (IOException e) {e.printStackTrace(); return; }  
  
        out.println( "<html><head></head><body><h1>Welcome, " );  
        out.println(userName); /* Schreiben in HTML-Senke */  
        out.println(" </h1></body></html>");  
  
    }  
  
}
```

Flowfiles:

javax.servlet.http.HttpServletRequest.flow

Im Beispielcodes wird aus unsicheren Quelle gelesen. Dies muss durch eine Flow-Datei beschrieben werden. Die Flow-Datei sieht wie folgt aus:

```

summary HttpServletRequest : ServletRequest {

    http;
    parameter;
    parameterName;

    /* Initialisierung */
    {
        this.http ~ source('HTTP')      /* HTTP-Quelle Definition */
        this.parameter <* String(this.http)  /* Parameter liest aus
                                           HTTP-Quelle */

        this.parameterName <* String(this.http)
        ...
    }

    ...

    String getParameter(String parametername) {
        <- this.parameter                /* Datenfluss aus HTTP-Quelle */
    }
}

```

javax.servlet.http.HttpServletResponse.flow

Diese Flow-Datei beschreibt die Datenflüsse in der Klasse HttpServletResponse. Es wird ein Writer zurückgeliefert, der eine HTML-Senke darstellt. Im folgenden Flow wird die HTML-Senke definiert und der entsprechende Writer damit verknüpft.

```

summary HttpServletResponse : ServletResponse {

http;
html;
writer;                                /* aus ServletResponse.flow */

{
    this.http ~ sink('HTTP')
    this.html ~ sink('HTML')           /* html ist eine HTML-Senke */
    this.writer <- [PrintWriter]
    this.writer.print ~ this.html      /* der zurückgegebenen Writer
                                        schreibt in die HTML Senke */
}

/* aus ServletResponse.flow */
PrintWriter getWriter() {              /* Der oben definierte Writer
                                        wird zurückgeliefert */

    <- this.writer
}

}

```

Report:

Im folgenden Report werden die Quelle (Source Overview) und die Senke (Sink Overview) sowie die Anzahl der kritischen Pfade (Path Overview) richtig erkannt und protokolliert. Die Quelle befindet sich in der Zeile 21 und die Senke in 29. Es gibt nur einen kritischen Pfad.

Analysis Report

App2.javaict.icc

Security Metrics

Metric ID	Metric Description	Actual Found	OK	I	W	E	S	Metric Grade
1002	Number of Vulnerable Paths	1	0	0	1	2	-	W
1003	Number of Sources	1	0	1	10	100	-	I
1004	Number of Sinks	1	0	1	10	100	-	I
1005	Number of Sanitizers	0	0	1	10	100	-	OK
1010	Maximum Path Length	3	-	-	-	-	-	OK
1015	Average Path Length	3	-	-	-	-	-	OK
1020	Affected Lines of Code	2	1	5	10	100	-	OK

Path Overview

Path-ID	Length	Begin	End	Rank
1	3	00021	00029	1000

Source Overview

Source-ID	Path-ID	Line	Expression
3098834048 [1-1]	1	00021	request.getParameter("userName")

Sink Overview

Sink-ID	Path-ID	Line	Expression
3098834112 [1-3]	1	00029	userName

Annotated Source Code

Source #1
App2.java
ict.icc

```

00001 package ict.ict;
00002
00003 import java.io.IOException;
00004 import java.io.PrintWriter;
00005 import java.sql.Connection;
00006 import java.sql.DriverManager;
00007 import java.sql.Statement;
00008 import java.sql.ResultSet;
00009
00010 import javax.servlet.http.HttpServlet;
00011 import javax.servlet.http.HttpServletRequest;
00012 import javax.servlet.http.HttpServletResponse;
00013 import javax.servlet.http.HttpSession;
00014
00015 /** Cross Site Scripting **/
00016 public class App2 extends HttpServlet {
00017
00018     protected void doGet( HttpServletRequest request, HttpServletResponse response )
00019     {
00020         /* Hier wird aus unsicheren HTTP-Quelle gelesen */
00021         String userName = request.getParameter( "userName" );

```

Source-ID	Path-ID	Expression
3098634048 [1-1]	1	request.getParameter("userName")

Element-ID	Path-ID	Expression
3098634112 [1-2]	1	userName

```

00022
00023     PrintWriter out = null;
00024
00025     try { out = response.getWriter(); }
00026     catch (IOException e) { e.printStackTrace(); return; }
00027
00028     out.println( "<html><head></head><body><h1>Welcome, " );
00029     out.println(userName);

```

Sink-ID	Path-ID	Expression
3098634112 [1-3]	1	userName

```

00030     out.println(" </h1></body></html>");
00031
00032     }
00033 }

```

7 Glossar

Assembly	Übersetzte Programmklassen werden als ausführbare Programme in sogenannten Assemblies zusammengefasst und bereitgestellt (vergleichbar mit Jar-Dateien in Java). Diese haben typischerweise die altbekannten Endungen .exe oder .dll, sind intern jedoch völlig anders strukturiert.
Basic Block	Code, welcher nur einen Einsprung- und einen Ausgangspunkt hat und keine weiteren Sprungbefehle enthält. Der Basic Block ist meist die Basiseinheit, welche bei der Compiler Optimierung zur Anwendung kommt. Basic Blocks bilden die Eck- oder Knotenpunkte innerhalb des Kontrollflussgraphen.
BDD	Binary Decision Diagram: Binärer Entscheidungsbaum, bei dem in einer Abfolge von binären, hierarchischen Entscheidungen ein Ergebnis gesucht wird. BDDs eignen sich gut um die charakteristische Funktion einer Menge darzustellen.
BDDBDDDB	Binary Decision Diagram Based Deductive Database: Eine deduktive Datenbank mit Regelinterpreter. Relationen werden nicht als Tupelmengen dargestellt sondern als BDDs. Relationale Operatoren werden nicht als Mengenoperationen sondern als logische Operationen durchgeführt.
Boxing	Die Konvertierung eines Typs in einen anderen, allgemeineren Typ. Beispielsweise das Verpacken von primitiven int-Werten in Integer-Typen.
Buffer Overflow	Ein Pufferüberlauf wird erreicht, wenn die Daten, die in einen Puffer geschrieben werden, größer sind als der Puffer. Unter Umständen kommt es dabei zu einem Überschreiben der dem Puffer folgenden Speicherinhalte.
Buffer Overrun	siehe Buffer Overflow
CAG	siehe Callgraph
Callgraph	Die Knoten des Callgraphs sind die Methoden und Aufrufstellen des betrachteten Systems. Die Kanten des Callgraphs verbinden Aufrufstellen mit den Aufgerufenen Methoden (oder im Falle virtueller Aufrufe Methodenmengen)
CFG	<i>siehe Control Flow Graph</i>
Control Flow Graph	Control Flow Graph: Kontrollflussgraph, der zur Darstellung aller möglichen Pfade genutzt wird, die beim Ablauf eines Programmes beschriftet werden können. Sie werden unter anderem zur Programmoptimierung eingesetzt. Den Wurzelknoten kann man sich als Startpunkt des Computer-Programmes, den Exit-Knoten als seinen Endpunkt vorstellen. Wenn von einem Knoten mehrere Kanten wegführen (der Knoten also Quelle mehrerer gerichteter Kanten ist), so entspricht das einer Verzweigung. Schleifen finden sich als Zyklen in Kontrollflussgraphen wieder.
CFG Navigator	Visitor über den CFG, traversiert den CFG hierarchisch und führt je nach Instrumentierung verschiedene Aktionen durch.
CIL	Common Intermediate Language: Früher Microsoft Intermediate Language. Eine von Menschen lesbare Zwischensprache, zu der die Sprachen des .Net-Frameworks kompiliert werden und aus der der Bytecode abgeleitet wird.

CLI	Common Language Infrastructure: Die Infrastruktur unter den Microsoft Sprachen des .Net-Frameworks und Mono. Beschreibt einen zweistufigen Prozess, bei dem eine Ausgangssprache wie C# in die Common Intermediate Language kompiliert wird und von dort aus über die Laufzeitumgebung in ausführbare Programme gewandelt wird.
CLR	Common Language Runtime: Eine Laufzeitumgebung ähnlich der Java Virtual Machine, die in in Common Intermediate Language geschriebenen Code ausführbar macht.
CLS	Common Language Specification: Eine Spezifikation, die beschreibt, welche Eigenschaften eine Sprache haben muß, um nach CIL kompiliert werden zu können.
CTS	Common Type System: Eine allgemeine Spezifikation, wie Typen im Speicher definiert und repräsentiert werden sollen. Dient der Interoperabilität verwandter Sprachen, die sich ihr unterwerfen.
Code Cache	Der Quad-Code wird unter Umständen erst On-the-Fly erzeugt, der Code Cache cached diesen erzeugten Code für spätere Verwendung
Constraint	Bedingungen und Einschränkungen. Häufig im Zusammenhang von Eingabevalidierungen genutzt, bei denen Constraints die Grenzen der Werte darstellen.
Context	Eine inhaltlich zusammenhängende Sammlung von Daten, die einen Zustand in einem Ablauf darstellen.
Datalog	Datalog ist eine Datenbank-Programmiersprache, die Prolog syntaktisch und semantisch ähnelt.
Datalog Relation	Datalog Prädikat
Delegate	In C# kann man Delegate-Variablen wie Methodenzeiger verwenden. Zusätzlich unterstützen C#-Delegates weitere Operationen. Auch in Objective-C werden Delegates verwendet. Sie werden verwendet, um vom Delegierer vordefinierte Mitteilungen (Notifications) an ein Delegate zu senden. A reference type such that an instance of it can encapsulate one or more methods in an invocation list. Given a delegate instance and an appropriate set of arguments, one can invoke all of the methods in a delegate's invocation list with that set of arguments.
Dominator Node	Knoten in einem Kontrollflußgraphen, durch den alle Relationen zu einem dominierten Knoten führen.
Dominator Frontier	Die Grenzknoten in einem Kontrollflußgraphen, bis zu der sich Dominanzrelationen erstrecken.
ECMA 335	Spezifikation der Common Language Infrastructure (CLI)
Exception	Ausnahmezustand, meist genutzt im Hinblick auf unvorhergesehene Systemzustände (Fehler) innerhalb eines Programms.
False Positive	Eine fälschlich gemachte positive Aussage, in diesem Kontext ein sicherer Datenfluß, der als unsicher erkannt wird
Flow	Beschreibungssprache zur Spezifikation von Datenflüssen.
Guard	Konstruktion deren Kontrollfluß ausschließt, dass unsichere Daten über sie hinweg fließen können.
Intermediate Code	Zwischencode, der aus einer anderen Programmiersprache abgeleitet wird und zur Kompatibilität mit einem Compiler oder einer Laufzeitumgebung dient.
Kante (CFG)	Repräsentiert einen "Sprung" im Kontrollfluß

Liveness Analysis	Code Analyse, bei der festgestellt wird, wie häufig und wie lang eine Variable genutzt wird, meist um damit Optimierungen durchzuführen und ungenutzten Code zu identifizieren.
Location	Position im Quadcode
managed Data	verwaltete Daten mit Garbage Collection. Dereferenzierte Daten werden wieder freigegeben und zurückgesetzt
Mbel	Microsoft Bytecode Engineering Library. API zum Parsen, Editieren und Erstellen von .Net-Dateien.
Mono	Eine .Net-kompatible Entwicklungs- und Laufzeitumgebung, die auf der CLI basiert und .Net auf Unix-artigen Systemen lauffähig macht.
Null Check	Prüfung eines Wertes auf null, damit wird das Vorhandensein eines Wertes getestet
.NET	Eine Software-Plattform von Microsoft, die eine Laufzeitumgebung, verschiedene APIs und Dienstprogramme umschließt. Unterstützt momentan die Sprachen ASP.Net, Visual Basic, C# und J#.
Pfadelement	Stellen im verwundbaren Pfad, die von den Eingabedaten auf ihrem Weg von der Quelle zur Senke durchlaufen werden. Quelle und Senke selbst sind auch Pfadelemente.
Points-To Analyse	Points-To-Analyse ist eine statische Programmanalyse, die potentielle Beziehungen zwischen Pointern und Daten eines Programms berechnet.
PQL	Program Query Language
Predecessor	Vorgänger Basic Block im Kontrollflussgraph
Primitive	Einfacher, elementarer Datentyp
Prolog	Eine Programmiersprache, die als Maschinensprache eines Logik-Prozessors bezeichnet wird, da sie auf den mathematischen Grundlagen der Prädikatenlogik beruht. Ein Prolog-Programm ist eine Sammlung von so genannten Horn-Klauseln.
Reaching Definition Analysis	Diese Form der Analyse beschreibt welche Wertdefinitionen eine bestimmte Wertdefinition im Programmablauf unmittelbar erreichen können.
Reflection	Aufruf und Instantiierung von Klassen und Methoden über ihre jeweilige Bezeichnung zur Laufzeit.
Reflective Call	Reflektiver Aufruf einer Klasse und/oder einer Methode.
Register Factory	Factory Objekt, das Stackregister verwaltet, also anlegt und zur Verfügung stellt.
Register Graph	Graphen-Darstellung der Register.
Reverse Post Order	Umdrehen der Ordnung, nach der die Objekte eingegeben wurde.
Rule	Hier: Datalog-Regel
Runtime Analysis	Laufzeitanalyse
Sanitizer	Methoden die unabhängig von ihrer Effektivität, Daten um gefährliche Eingaben bereinigen.
Signature	Die Signatur ist eine Beschreibung der Schnittstellen von Methoden oder Klassen.

Sink	Eine Senke aus Sicht des betrachteten Programms ist jede Operation, bei der Daten aus dem Programm einen sicherheitskritischen Ablauf außerhalb des Programms beeinflussen. Typischerweise sind dies Operationen deren Daten als Eingabeparameter bestimmter Systemfunktionen, z.B. von Betriebssystem-Kommandozeilenaufrufen, Datenbankoperationen, oder zu Manipulationen externer Prozesse dienen.
Source	Hier: Eine Quelle aus Sicht des betrachteten Programms ist jede Operation, durch die Daten von außerhalb des Programms in das Programm hinein gelangen - also jede Art von üblichen Eingabeoperationen (z.B. Empfang von Daten aus einer Netzwerkverbindung), aber auch alle Zugriffe auf externe Aufrufparameter, Umgebungsvariable, das Dateisystem etc. Quellen können Teil der Programmiersprache sein (z.B. die Aufrufparameter der main()-Funktion in C), der Standardbibliothek der Sprache (z.B. die scanf()-Funktion in C) angehören, oder verschiedensten Bibliotheken zum Zugriff auf das Betriebssystem, auf Middleware, oder auf anwendungsspezifische Funktionen entstammen
SQL Injection	SQL-Injektion (engl. SQL Injection) bezeichnet das Ausnutzen einer Sicherheitslücke in Zusammenhang mit SQL-Datenbanken, die durch mangelnde Maskierung oder Überprüfung von Metazeichen in Benutzereingaben entsteht. Der Angreifer versucht dabei über die Anwendung, die den Zugriff auf die Datenbank bereitstellt, eigene Datenbankbefehle einzuschleusen. Sein Ziel ist es dabei, Daten in seinem Sinne zu verändern oder Kontrolle über den Server zu erhalten.
SSA	Static Single Assignment. Spezielle Klasse von Zwischencode. Sie zeichnet sich dadurch aus, dass im Zwischencode jede Variable statisch nur einmal zugewiesen wird. Dadurch werden Datenabhängigkeiten zwischen Befehlen explizit dargestellt, was für viele Optimierungen von Vorteil ist. Die Quellprogramme vieler Programmiersprachen lassen sich ohne größeren Aufwand in eine SSA Darstellung transformieren. Viele Moderne Compiler - darunter der GNU C Compiler - verwenden daher SSA basierten Zwischencode.
Static Analysis	Statische Analyse. Hier: Die statische Analyse ist ein Softwaretest, bei dem bestimmte Eigenschaften (hier: Schwachstellen) anhand des vorliegenden Quellcodes erkannt werden können.
Static Call	Statischer Methodenaufruf. Aufruf einer statischen Methode ohne Kenntnis der Objektinstanz
Static Initializer	Statische Initialisierer sind Code Blöcke, die ausgeführt werden, wenn statische Felder deklariert werden.
Successor	Nachfolge Basic Block im Kontrollflussgraph
Summary File	Datei, in der ein oder mehrere Tupel mit Datenflußbeschreibungen (FLOWS) definiert werden.
Summary Parser	Parser für Summary Files.
Temporary Register	Temporäres Register
Tupel	Geordnete Zusammenstellung von Objekten, deren Elemente eine festgelegte Reihenfolge haben (im Gegensatz zur ungeordneten Menge). (a,b,c)
unmanaged data	Unverwaltete Daten ohne Garbage Collection - referenzierte Daten werden nach der Dereferenzierung nicht wieder automatisch freigegeben

Validierung	Die Validierung von Daten stellt sicher, dass die zunächst nicht vertrauenswürdigen Daten aus den externen Quellen auf ihre Zulässigkeit überprüft werden, bevor sie - in der Regel indirekt über eine verkettete Beeinflussung beliebig vieler anderer Datenobjekte - in eine Senke einfließen dürfen. Ein Datenobjekt ist valide an einem bestimmten Punkt in der Programmausführung, falls es entweder aus einer vertrauenswürdigen Quelle stammt (konstante Daten sind z.B. immer valide) oder irgendwann vor diesem Punkt im oben beschriebenen Sinne validiert wurde. Daten, die - direkt oder indirekt - aus einer nicht vertrauenswürdigen Quelle stammen, sind damit invalid, und auch das Ergebnis der Verknüpfung eines validen Operanden mit einem invaliden Operanden ist selbst wieder invalid.
Value Type	Wertetyp.
Virtual Call	Virtueller Methodenaufruf.
Vulnerable path	Sicherheitskritischer Datenfluss (siehe Vulnerability)
Vulnerability	Schwachstelle. Hier: Abfolge von Methoden und Zuweisungen, die es erlauben, daß ungefilterte Daten von externen Quellen an sicherheitsrelevante Senken weitergereicht werden.
Wrapper	Verpackung. Hier: Klassendefinition, die eine andere Klasse einschließt, um beispielsweise eine andere Schnittstelle für die Klasse bieten zu können.
XSB Prolog	Weit verbreitete Prolog Variante
XSS	Cross-Site Scripting (XSS) bezeichnet das Ausnutzen einer Computersicherheitslücke, indem Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext eingefügt werden, in dem sie als vertrauenswürdig eingestuft sind. Aus diesem vertrauenswürdigen Kontext kann dann ein Angriff gestartet werden. Die Bezeichnung Cross-Site leitet sich von der Art ab, wie diese Attacke webseitenübergreifend ausgeführt wird (auf einer vom Angreifer kontrollierten Seite steht beispielsweise ein präparierter Hyperlink, der zur vermeintlich vertrauenswürdigen Website einer meist ahnungslosen dritten Partei führt). Cross-Site Scripting wird manchmal auch CSS abgekürzt, hat jedoch nichts mit der Cascading Style Sheet-Technologie zu tun, die weit häufiger CSS genannt wird. Um Verwechslungen zu vermeiden, sollte daher die Abkürzung XSS benutzt werden.
Zwischencode	Siehe Intermediate Code

8 Literatur- und Linksammlung

- [1] S. Ceri, G. Gottlob and L. Tanca: Logic Programming and Databases, Springer-Verlag, Surveys in Computer Science, Berlin, Heidelberg, New York, 1990.
- [2] C.J. Date: An Introduction to Database Systems, 6th ed., Addison-Wesley, 1990.
- [3] R. Elmasri and S. B. Navathe: Fundamentals of Database Systems, 2nd ed., Benjamin/Cummings, 1994
- [4] M. Lam, M. Martin, B. Livshits, J. Whaley: Securing Web Applications Using Static and Dynamic Information Flow Tracking. In ACM Sigplan 2008 Workshop on Partial Evaluation and Program Manipulation, San Francisco, CA, January 2008.
- [5] M. S. Lam, J. Whaley, V. B. Livshits, M. Martin, D. Avots, M. Carbin, C. Unkel: Context-Sensitive Program Analysis as Database Queries. In Proceedings of Principles of Database Systems (PODS), Baltimore, Maryland, June 2005.
- [6] B. Livshits, J. Whaley, M. S. Lam: Reflection Analysis for Java. In Proceedings of Programming Languages and Systems: Third Asian Symposium, APLAS 2005, Tsukuba, Japan, November 2005.
- [7] J. Whaley. Context-Sensitive Pointer Analysis using Binary Decision Diagrams. Ph.D. thesis, Stanford University, March 2007.
- [8] J. Whaley. Joeq: A Virtual Machine and Compiler Infrastructure. In Science of Computer Programming Journal, 2005.
- [9] J. Whaley: Joeq: A Virtual Machine and Compiler Infrastructure. In Proceedings of the Workshop on Interpreters, Virtual Machines, and Emulators, San Diego, CA, June 2003. Pages 58-66.
- [10] J. Whaley, D. Avots, M. Carbin, M. S. Lam: Using Datalog with Binary Decision Diagrams for Program Analysis. In Proceedings of Programming Languages and Systems: Third Asian Symposium, APLAS 2005, Tsukuba, Japan, November 2005.
- [11] J. Whaley, M. S. Lam: Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams. In Proceedings of Program Language Design and Implementation (PLDI), Washington DC, June 2004.

Joeq, bddbddb: <http://suif.stanford.edu/~jwhaley/>

pbd2xml: <http://blogs.msdn.com/jmstall>

Jasper/iReport: www.jasperforge.org

Quad: <http://cycleserv2.csail.mit.edu/Harpoon/quads/quads.pdf>

Quad: http://suif.stanford.edu/~courses/cs243/joeq/quad_overview.html

SecFlow: www.secflow.org

SecFlow: www.sourceforge.net

Datei	Version	Datum	Bearbeiter/in	Seite
SecFlow_Benutzerhandbuch_V10.odt	1.0	2008-11-27	JK/DR	63 von 67